| **Committee Draft ISO/IEC CD2 11179-3** | |
|---|---|
| Date:<br>**2009-03-22** | Reference number:<br>ISO/JTC 1/SC 32**N1851** |
| Supersedes document SC 32N1667 | |

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

| | |
|---|---|
| ISO/IEC JTC 1/SC 32<br>Data Management and Interchange<br><br>Secretariat:<br>USA (ANSI) | Circulated to P- and O-members, and to technical committees and organizations in liaison for voting (P-members only) by:<br><br>**2009-06-22**<br><br>Please return all votes and comments in electronic form directly to the SC 32 Secretariat by the due date indicated. |

ISO/IEC CD2 11179-3: 2009(E)

Title: Information technology -- Metadata Registries (MDR) - Part 3: Registry Metamodel and basic attributes, 3rd Edition

Project: 1.32.15.03.03.00

Introductory note:

The attached document is hereby submitted for a 3-month letter ballot to the NBs of ISO/IEC JTC 1/SC 32. The ballot starts 2009-03-22. The disposition of comments on 32N1667 CD 11179-3 is 32N1852.

Medium: E

No. of pages: 208

Dr. Timothy Schoechle, Secretary, ISO/IEC JTC 1/SC 32
Farance Inc *, 3066 Sixth Street, Boulder, CO, United States of America
Telephone: +1 303-443-5490; E-mail: Timothy@Schoechle.org
available from the JTC 1/SC 32 WebSite http://www.jtc1sc32.org/
*Farance Inc. administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

**ISO/IEC JTC 1/SC 32 N 1851**

Date: 2009-03-23

**ISO/IEC CD 11179-3.2**

ISO/IEC JTC 1/SC 32/WG 2

Secretariat: ANSI

# Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes

*Technologies de l'information — Registres de métadonnées (RM) — Partie 3: Métamodèle de registre et attributs de base*

**Warning**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International Standard
Document subtype:
Document stage: (30) Committee
Document language: E

C:\Documents and Settings\gatesr\My Documents\SC32 docs\32N1851T-text_for_ballot-CD2_11179-3.doc STD Version 2.0

# Contents

Page

Foreword ..... ix

Introduction ..... x

1 Scope ..... 1
1.1 Overview ..... 1
1.2 Scope – Structure of a Metadata Registry ..... 1
1.3 Scope – Basic attributes of metadata items ..... 1
1.4 Summary of changes from Edition 2 ..... 1
1.5 Relationship to other parts of this International Standard ..... 2

2 Normative references ..... 2

3 Definitions ..... 3
3.1 Overview ..... 3
3.2 Definitions of Metamodel Constructs ..... 3
3.3 Broad Terms used in this part of ISO/IEC 11179 ..... 5
3.4 Terms used in the specification of the metamodel ..... 9
3.5 List of Abbreviations and Acronyms ..... 18

4 Structure of a Metadata Registry ..... 21
4.1 Metamodel for a Metadata Registry ..... 21
4.2 Application of the metamodel ..... 21
4.3 Specification of the metamodel ..... 22
4.3.1 Terminology used in specifying the metamodel ..... 22
4.3.2 Use of UML Packages ..... 22
4.3.3 Package Dependencies ..... 23
4.3.4 Use of UML Class diagrams and textual description ..... 23
4.4 Types, Instances and Values ..... 23
4.5 Types of Items in an ISO/IEC 11179 metadata registry ..... 24
4.5.1 Rules for Identified_Item and its subtypes. ..... 25
4.6 Extensibility ..... 25
4.7 Date References ..... 26

5 Basic Package ..... 27
5.1 Basic types and classes metamodel region ..... 27
5.1.1 Overview ..... 27
5.1.2 Boolean ..... 27
5.1.3 Contact ..... 27
5.1.4 Date-and-time ..... 29
5.1.5 Individual ..... 29
5.1.6 Integer ..... 29
5.1.7 Language_Identification ..... 30
5.1.8 Natural_Range ..... 32
5.1.9 Notation ..... 32
5.1.10 Organization ..... 32
5.1.11 Phone_Number ..... 33
5.1.12 Postal_address ..... 33
5.1.13 Reference_Document ..... 33
5.1.14 Registration_Authority_Identifier ..... 35
5.1.15 Sign ..... 36
5.1.16 String ..... 36
5.1.17 Text ..... 36

5.1.18 Value ..... 36

6 Identification, Designation and Definition Package ..... 38
6.1 Identification region ..... 38
6.1.1 Overview ..... 38
6.1.2 Classes in the Identification metamodel region ..... 38
6.1.3 Associations in the Identification region ..... 43
6.2 Designation and Definition region ..... 44
6.2.1 Overview ..... 44
6.2.2 Classes in the Designation and Definition region ..... 45
6.2.3 Association Classes in the Designation and Definition Region ..... 52
6.2.4 Associations in the Designation and Definition Region ..... 53

7 Registration Package ..... 54
7.1 Registration metamodel region ..... 54
7.1.1 Overview ..... 54
7.1.2 Classes in the Registration region ..... 56
7.1.3 Classes referenced from the Basic package ..... 66
7.1.4 Classes referenced from the Identification, Designation and Definition package ..... 67
7.1.5 Association Classes in the Registration region ..... 67
7.1.6 Associations in the Registration region ..... 68

8 Concepts Package ..... 69
8.1 Concept System region ..... 69
8.1.1 Overview ..... 69
8.1.2 Classes in the Concept_System region ..... 70
8.1.3 Associations of the Concept_System region ..... 74
8.1.4 Association Classes in the Concept_System Region ..... 76
8.2 Classification metamodel region ..... 77
8.2.1 Overview ..... 77
8.2.2 Classes in the Classification region ..... 78
8.2.3 Associations Classes in the Classification Region ..... 78
8.2.4 Associations in the Classification Region ..... 78

9 Binary_Relations Package ..... 79
9.1 Binary_Relations metamodel region ..... 79
9.1.1 Overview ..... 79
9.1.2 Classes in the Binary_Relations metamodel region ..... 79

10 Data Description Package ..... 82
10.1 High-level Data Description metamodel ..... 82
10.1.1 Overview ..... 82
10.1.2 Classes of High-level Data Description Metamodel ..... 83
10.1.3 Associations of the High Level Metamodel ..... 85
10.1.4 Constraints of the High Level Metamodel ..... 86
10.2 Data Element Concept region ..... 87
10.2.1 Overview ..... 87
10.2.2 Classes in the Data_Element_Concept region ..... 87
10.2.3 Associations in the Data_Element_Concept region ..... 88
10.3 Conceptual and Value_Domain region ..... 89
10.3.1 Overview ..... 89
10.3.2 Classes in the Conceptual and Value_Domain region ..... 91
10.3.3 Associations in the Conceptual and Value_Domain region ..... 97
10.3.4 Additional Constraints of the Conceptual and Value_Domain region ..... 98
10.4 Measurement region ..... 99
10.4.1 Overview ..... 99
10.4.2 Classes in the Measurement region ..... 100
10.4.3 Associations in the Measurement region ..... 102
10.5 Data_Element region ..... 102
10.5.1 Overview ..... 102
10.5.2 Classes in the Data_Element Region ..... 103
10.5.3 Associations in the Data_Element region ..... 106

**10.6 Consolidated Data Description Metamodel ..... 107**
**10.7 Types of Concepts in the Data Description Metamodel ..... 108**

**11 Basic attributes ..... 109**
**11.1 Use of basic attributes ..... 109**
**11.2 Common attributes ..... 109**
**11.2.1 Identifying ..... 109**
**11.2.2 Definitional ..... 110**
**11.2.3 Administrative ..... 111**
**11.2.4 Relational ..... 111**
**11.3 Attributes specific to Data_Element_Concepts ..... 111**
**11.4 Attributes specific to Data_Elements ..... 112**
**11.5 Attributes specific to Conceptual_Domains ..... 112**
**11.6 Attributes specific to Value_Domains ..... 112**
**11.7 Attributes specific to Permissible_Values ..... 113**
**11.8 Attributes specific to Value_Meanings ..... 113**

**12 Conformance ..... 114**
**12.1 Overview of Conformance ..... 114**
**12.2 Degree of Conformance ..... 114**
**12.2.1 Strictly conforming implementations ..... 114**
**12.2.2 Conforming implementations ..... 115**
**12.3 Conformance by Clause ..... 115**
**12.4 Registry Conformance ..... 115**
**12.4.1 Overview ..... 115**
**12.4.2 Standard Profiles for Edition 3 Registries ..... 115**
**12.4.3 Conformance Levels for Edition 2 Level 2 ..... 116**
**12.4.4 Conformance Levels for Edition 2 Level 1 and Edition 1 ..... 116**
**12.5 Obligation ..... 116**
**12.6 Implementation Conformance Statement (ICS) ..... 116**
**12.7 Roles and Responsibilities for Registration ..... 117**

**Annex A** (normative) **Alphabetical List of Terms ..... 118**

**Annex B** (normative) **Consolidated Class Hierarchy ..... 125**

**Annex C** (informative) **Alternative representation of the metamodel ..... 126**
**C.1 Major Item Elements ..... 127**
**C.2 Major Item Elements Detail ..... 128**
**C.3 Supporting Elements ..... 129**
**C.4 Supporting Elements Detail ..... 130**
**C.5 Definition & Designation Elements ..... 131**
**C.6 Definition & Designation Elements Detail ..... 132**
**C.7 Item Registration Elements ..... 133**
**C.8 Item Rules ..... 134**
**C.9 Item Registration Elements Detail ..... 135**
**C.10 Data Description ..... 136**
**C.11 Concept ..... 137**
**C.12 Concept Detail ..... 138**
**C.13 Data Element & Value Domain ..... 139**
**C.14 Data Element & Value Domain Detail ..... 140**

**Annex D** (informative) **Mapping the ISO/IEC 11179-3:1994 basic attributes to the ISO/IEC 11179-3:200n metamodel and basic attributes ..... 141**
**D.1 Introduction ..... 141**
**D.1.1 Description of Table Structures in this Annex ..... 142**
**D.2 Mapping the Basic Attributes ..... 145**
**D.2.1 Common Identifying attributes ..... 145**
**D.2.2 Common Definitional attributes ..... 151**
**D.2.3 Common Administrative attributes ..... 152**
**D.2.4 Common Relational attributes ..... 155**
**D.2.5 Attributes specific to Data_Element_Concepts ..... 159**
**D.2.6 Attributes specific to Data_Elements ..... 161**

**D.2.7 Attributes specific to Conceptual_Domains ..... 168**
**D.2.8 Attributes specific to Value_Domains ..... 169**
**D.2.9 Attributes specific to Permissible_Values ..... 170**
**D.2.10 Attributes specific to Value_Meanings ..... 171**

**Annex E** (informative) **Mapping the ISO/IEC 11179-3:2002 metamodel to the ISO/IEC 11179-3:200n metamodel ..... 172**
**E.1 Introduction ..... 172**
**E.2 Mapping the Edition 2 Common Facilities ..... 172**
**E.3 Mapping the Data Description Model ..... 172**

**Annex F** (informative) **Concept System Examples ..... 173**
**F.1 Concept System Metamodels ..... 173**
**F.2 SKOS Example ..... 174**
**F.2.1 SKOS Metamodel ..... 174**
**F.2.2 SKOS Example Thesaurus ..... 174**
**F.2.3 Example Value Domain References ..... 176**
**F.3 ORM Example ..... 177**
**F.3.1 ORM Metamodel ..... 177**
**F.3.2 Car Registration Model ..... 179**
**F.4 OWL Example ..... 183**
**F.4.1 OWL Metamodel ..... 183**
**F.4.2 Car Registration Ontology ..... 189**

**Bibliography ..... 195**

# Table of Figures

Figure 4-1 — Package dependencies ..... 23

Figure 4-2 — Types of items ..... 24

Figure 5-1 — Basic types and classes metamodel region ..... 27

Figure 6-1 — Identification metamodel region ..... 38

Figure 6-2 — Designation and Definition metamodel region ..... 45

Figure 7-1 — Registration metamodel region ..... 55

Figure 7-2 — Registration_Record ..... 56

Figure 7-3 — Registry specification ..... 56

Figure 8-1 — Concept system metamodel region ..... 69

Figure 8-2 — Classification metamodel region ..... 77

Figure 9-1 — Binary Relations metamodel region ..... 79

Figure 10-1 — High-level Data Description metamodel ..... 83

Figure 10-2 — Data_Element_Concept metamodel region ..... 87

Figure 10-3 — Conceptual and value domain metamodel region ..... 90

Figure 10-4 — Measurement metamodel region ..... 100

Figure 10-5 — Data_Element metamodel region ..... 103

Figure 10-6 — Consolidated Data Description metamodel ..... 107

Figure 10-7 — Types of Concepts in the Data Description package ..... 108

Figure B-1 — Consolidated Class Hierarchy ..... 125

Figure D.-1 — Basic Attributes of Data elements ..... 142

Figure F-1 — Car Registration Model in ORM ..... 179

Figure F-2 — Car Registration Ontology ..... 189

# Table of Tables

Table 4-1: Rules for Types of Items ..... 25

Table 6-1: Comparison of Designation to Scoped_Identifier ..... 46

Table 9-1: Examples of binary relations and their characterization ..... 80

Table 12-1 – Comparison for Conformance Levels across Editions of ISO/IEC 11179-3 ..... 115

Table F-1: Correspondences of 11179-3 concept system metamodel to selected notations ..... 173

Table F-2: SKOS-CORE as a 11179 Concept System ..... 174

Table F-3: SKOS relations as a 11179 Binary Relations ..... 174

Table F-4: SKOS Thesaurus Example – 11179 Concept System ..... 175

Table F-5: SKOS Thesaurus Example – 11179 Concepts ..... 175

Table F-6: SKOS Thesaurus Example – 11179 Links ..... 176

Table F-7: SKOS Thesaurus Example – 11179 Conceptual Domains ..... 176

Table F-8: SKOS Thesaurus Example – 11179 Value Domains ..... 176

Table F-9: ORM as a 11179 Concept System ..... 177

Table F-10: ORM Relations as 11179 Binary Relations ..... 177

Table F-11: ORM Roles as 11179 Relation Roles ..... 178

Table F-12: Car Registration Model - 11179 Concept System ..... 181

Table F-13: Car Registration Model - 11179 Concepts ..... 181

Table F-14: Car Registration Model - 11179 Binary Relations ..... 182

Table F-15: Car Registration Model - 11179 Links ..... 183

# Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 11179 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 11179-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 32, *Data Management and Interchange*.

This third edition cancels and replaces the second edition, which has been technically revised.

ISO/IEC 11179 consists of the following parts, under the general title *Information technology — Metadata registries (MDR)*:

EDITOR'S NOTE #1. (Action Required by FCD) For the 3rd edition of ISO/IEC 11179, it is expected that part 2 will be withdrawn, since part 3 has subsumed its content.

— *Part 1: Framework*

— *Part 2: Classification*

— *Part 3: Registry metamodel and basic attributes*

— *Part 4: Formulation of data definitions*

— *Part 5: Naming and identification principles*

— *Part 6: Registration*

# Introduction

EDITOR'S NOTE #2. (Informational) Throughout this Committee Draft, EDITOR'S NOTEs make reference to 'issues' that are either addressed or not addressed by this document. Details of these issues may be found on the WG2 Issue Management website at: http://issues.metadata-stds.org . To locate a specific issue, the generic format of the URL is: http://issues.metadata-standards.org/show_bug.cgi?id=221 where the number at the end is the issue number, without leading zeroes.

EDITOR'S NOTE #3. (Action required) There have been extensive changes from both the second edition of this standard, and from CD1 of the third edition. The whole document needs careful review and comment.

Data processing and electronic data interchange rely heavily on accurate, reliable, controllable and verifiable data recorded in databases. A prerequisite for correct and proper use and interpretation of data is that both users and owners of data have a common understanding of the meaning and representation of the data. To facilitate this common understanding, a number of characteristics, or attributes, of the data have to be defined. These characteristics of data are known as “metadata”, that is, “data that describes data”. This part of ISO/IEC 11179 provides for the attributes of data elements and associated metadata to be specified and registered as metadata items in a *Metadata Registry*.

The structure of a *Metadata Registry* is specified in the form of a conceptual data model. The *Metadata Registry* is used to keep information about data elements and associated concepts, such as “data element concepts”, “conceptual domains” and “value domains”. Generically, these are all referred to as “metadata items”. Such metadata are necessary to clearly describe, record, analyse, classify and administer data.

When considering data and metadata, it is important to distinguish between types of data/metadata, and instances of these types. Clause 5 through 10 of this part of ISO/IEC 11179 specify the types of metadata objects that form the structure of a *Metadata Registry*. A *Metadata Registry* will be populated with instances of these metadata objects (metadata items), which in turn define types of data, e.g. in an application database. In other words, instances of metadata specify types of application level data. In turn, the application database will be populated by the real world data as instances of those defined data types.

NOTE ISO/IEC 10027:1990 *Information technology — Information resource dictionary system (*IRDS) Framework and ISO/IEC TR 10032:2003 *Information technology* — Reference model for data management explain the concepts of different levels of modelling.

In this part of ISO/IEC 11179, clause 11 describes the basic attributes of metadata items for purposes where a complete *Metadata Registry* is not appropriate.

This part of ISO/IEC 11179 is of interest to information developers, information managers, data administrators, standards developers and others who are responsible for making data understandable and shareable. ISO/IEC 11179 has broad applicability across subject area domains and information technologies.

This part of ISO/IEC 11179 applies to activities including:

a) the definition, specification and contents of metadata registries, including interchanging or referencing among various collections of data elements;

b) the design and specification of application-oriented data models, databases and message types for data interchange;

c) the actual use of data in communications and information processing systems;

d) interchange or reference among various collections of metadata;

e) the registration and management of semantic artifacts that are useful for data management, data administration, and data analysis;

f) the interrelation and mapping of concept systems with other concept systems, e.g., to support efforts to converge on consistency through harmonization and vetting activities;

g) the interrelation of concept systems with data held in relational databases, XML databases, knowledgebases, text, and possibly graph databases deriving from natural language text understanding systems

h) the provision of services for semantic computing: Semantics Service Oriented Architecture, Semantic Grids, semantics based workflows, Semantic Web, etc.;

i) support for addressing semantic web considerations such as AAA (anyone can say anything about anything), non-unique names, and open world assumption;

j) the capture of semantics with more formal techniques (in addition to natural language) -- First Order Logic (e.g., Common Logic), Description Logics (such as OWL-DL);

k) support of Application Development and Maintenance;

l) support of data migration, data mediation;

m) support of portals, data marts, and data warehouses;

n) support of data grids and online transaction networks;

o) ontological reasoning with metadata;

p) ontology entry point for browsing and searching metadata registries;

q) capture of associations between the published identifiers used in the ontology(s), and the concepts registered in the registry;

r) support for Ontology-driven Data Translation;

s) support for data integration & data interoperation;

# Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes

## 1 Scope

### 1.1 Overview

The primary purpose of ISO/IEC 11179-3 is to specify the structure of a *Metadata Registry* (see subclause 1.2). ISO/IEC 11179-3 also specifies basic attributes which are required to describe metadata items, and which may be used in situations where a complete metadata registry is not appropriate (e.g. in the specification of other International Standards) (see subclause 1.3).

Subclause 1.4 describes changes from the previous edition of this standard. Subclause 1.5 describes the relationship of this part to other parts of ISO/IEC 11179. Subclause 1.6 provides examples of activities where ISO/IEC 11179-3 may be applied.

### 1.2 Scope – Structure of a Metadata Registry

This part of ISO/IEC 11179 specifies the structure of the a metadata registry in the form of a conceptual data model is in Clauses 5 through 10.

While the model diagrams are presented in UML notation, this part of ISO/IEC 11179 does not assume nor endorse any specific system environment, database management system, database design paradigm, system development methodology, data definition_language, command language, system interface, user interface, computing platform, or any technology required for implementation. This part of ISO/IEC 11179 does not directly apply to the actual use of data in communications and information processing systems.

### 1.3 Scope – Basic attributes of metadata items

This part of ISO/IEC 11179 also specifies basic attributes which are required to describe metadata items, and which may be used in situations where a complete *Metadata Registry* is not appropriate (e.g. in the specification of other International Standards). These basic attributes are described in Clause 9.

### 1.4 Summary of changes from Edition 2

ISO/IEC 11179-3 Edition 3 includes several enhancements to Edition 2, both in terms of the presentation of the metamodel, and its capabilities, as follows:

From a presentation perspective, these include:

1. Use of UML 2.0 instead of UML 1.4 to describe the metamodel;

2. Use of UML packages to show dependencies between various regions of the metamodel. (See 4.3.2 and 4.3.3.)

From a capability perspective, these include:

3. Introduction of different types of metadata items (see 4.5);

4. Support for registration of Concept Systems.

5. Finer-grained conformance options (see clause 12).

### 1.5 Relationship to other parts of this International Standard

A comprehensive *Metadata Registry* management function requires a set of rules and procedures. Some of these rules and procedures are set out in the Clauses and Annexes of this part. These are complemented by the other parts of ISO/IEC 11179, as follows:

a) the overall framework for this family of International Standards is specified in ISO/IEC 11179-1;

b) rules and guidelines for classifying metadata are in ISO/IEC 11179-2;

c) rules and guidelines for the formulation of definitions are in ISO/IEC 11179-4;

d) naming and identifying principles for metadata are in ISO/IEC 11179-5;

e) rules and guidelines for registering metadata are in ISO/IEC 11179-6.

## 2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IETF RFC 4646, *Tags for Identifying Languages*[1]

ISO 31-0, *Quantities and units — Part 0: General principles*

ISO 639-2:1998, *Codes for the representation of the names of languages — Part 2: Alpha-3 code*

ISO 1087-1:2000, *Terminology work — Vocabulary — Part 1: Theory and application*

ISO/IEC 2382-1:1993, *Information technology — Vocabulary — Part 1: Fundamental terms*

ISO/IEC 2382-17:1999, *Information technology — Vocabulary — Part 17: Databases*

ISO 3166-1:2006, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*

ISO 5127:2001, *Information and documentation — Vocabulary*

ISO/IEC 6523-1:1998, *Information technology — Structure for the identification of organization and organization parts — Part 1: Identification of organization identification schemes*

ISO/IEC 6523-2:1998, *Information technology — Structure for the identification of organization and organization parts — Part 2: Registration of organization identification schemes*

ISO 8601:2000, *Data elements and interchange formats — Information exchange — Representation of dates and times*

ISO 10241:1992, *International terminology standards — Preparation and layout*

ISO/IEC 11179-1, *Information technology — Metadata registries (MDR) — Part 1: Framework*

---

[1] IETF RFC 4646 is available at http://www.rfc-editor.org/rfc/rfc4646.txt

ISO/IEC 11179-6, *Information technology — Metadata registries (MDR) — Part 6: Registration*

ISO/IEC 11404:2007, *Information technology — General Purpose Datatypes (GPD)*

ISO 12620:1999, *Computer applications in terminology — Data categories*

ISO 15924:2004 *Information and documentation — Codes for the representation of names of scripts*

ISO/IEC DIS 19501-2:2008, *Information technology — Unified Modeling Language (UML) Version 2.1.2 — Part 2: Superstructure*

ISO/IEC 19773:200n, Information technology — Metadata registries (MDR) Modules

ITU-T Recommendation E.164 (2005-02) The international public telecommunications numbering plan[2]

Universal Postal Union (UPU) S42-1:2003 International postal address components and templates[3]

# 3 Definitions

## 3.1 Overview

For the purposes of this document, the terms and definitions listed below apply. They have been organized into sub-clauses as follows:

— 3.2 defines metamodel constructs used in specifying the registry metamodel;

— 3.3 lists broad terms, and their definitions, used in this document;

— 3.4 defines concepts represented by the metamodel that is specified in clauses 5 through 10;

— 3.5 lists abbreviations and acronyms used in this standard.

An alphabetical list of all terms used in the standard is provided in Annex A, including data definitions from clauses 5 through 10.

NOTE Some definitions listed in this clause have one or more NOTEs, some have a reference to another standard from which the definition is taken and some have both NOTEs and a reference. Where a definition has both NOTEs and a reference, NOTEs that precede the reference come from the referenced source; NOTEs that follow the reference are added by this standard.

## 3.2 Definitions of Metamodel Constructs

This subclause defines the metamodel constructs used in specifying the registry metamodel in Clauses 5 through 10.

**3.2.1**
**association**
⟨metamodel⟩ semantic **relationship** between two **classes**

NOTE An **association** is a type of **relationship**.

---

[2] ITU-T E.164 is available at http://www.itu.int/rec/T-REC-E.164-200502-I/en

[3] UPU is the Universal Postal Union at "http://www.upu.int". UPU S42-1 is based on EN 14142-1, Postal services – Address data bases – Part 1 – Components of Postal_Addresses.

[Adapted from ISO/IEC DIS 19501-2:2008, 7.3.3]

**3.2.2**
**association class**
⟨metamodel⟩ **association** that is also a **class**

NOTE It not only connects a set of **class**es, but also defines a set of features that belong to the **association** itself.

[Adapted from ISO/IEC DIS 19501-2:2008, 7.3.4]

**3.2.3**
**attribute**
⟨metamodel⟩ **characteristic** of an **object** or **entity**

**3.2.4**
**class**
⟨metamodel⟩ description of a set of **object**s that share the same **attribute**s, operations, methods, **relationship**s, and semantics

[Adapted from ISO/IEC DIS 19501-2:2008, 7.3.7]

**3.2.5**
**composite attribute**
⟨metamodel⟩ **attribute** whose **datatype** is non-atomic

**3.2.6**
**composite datatype**
⟨metamodel⟩ **datatype** that is also a **class**

NOTE A **composite datatype** is used as a **datatype** for a **composite attribute**.

**3.2.7**
**datatype**
set of distinct values, characterized by properties of those values and by operations on those values

[ISO/IEC 11404:2007, 3.12]

**3.2.8**
**generalization**
⟨metamodel⟩ **relationship** between a more general **class** (the parent) and a more specific **class** (the child) that is fully consistent with the first **class** (i.e. it has all of its **attributes** and **relationships**) and that adds additional information.

NOTE A **generalization** is a type of **relationship**.

[Adapted from ISO/IEC DIS 19501-2:2008, 7.3.20]

**3.2.9**
**identifier**
⟨metamodel⟩ sequence of characters, capable of uniquely identifying that with which it is associated, within a specified context

NOTE A **name** should not be used as an identifier because it is not linguistically neutral.

**3.2.10**
**primitive datatype**
datatype that cannot be decomposed into other datatypes without loss of associated semantics.

[Adapted from ISO/IEC 11404:2007, 3.44]

**3.2.11**
**relationship**
⟨metamodel⟩ connection among model elements

NOTE In ISO/IEC 11179-3, a relationship is either an **association** or a **generalization**.

[Adapted from ISO/IEC 19501-2:2008, 7.3.47]

## 3.3 Broad Terms used in this part of ISO/IEC 11179

**3.3.1**
**acceptability rating**
scale of acceptability comprised of: preferred, admitted, deprecated, obsolete and superseded.

[ISO 10241:1992]

**3.3.2**
**attribute instance**
specific instance of an **attribute**

NOTE Amended from ISO 2382-17:1993 (17.02.13) to distinguish an instance of an attribute from its value.

**3.3.3**
**attribute value**
value associated with an **attribute instance**

NOTE Amended from ISO 2382-17:1993 (17.02.13) to distinguish an instance of an attribute from its value.

**3.3.4**
**basic attribute**
<metadata> **attribute** of a **metadata item** commonly needed in its specification

**3.3.5**
**binding**
mapping from one framework or specification to another, enabling **data** and/or commands to be passed between them

**3.3.6**
**characteristic**
abstraction of a property of an **object** or of a set of objects

NOTE Characteristics are used for describing **concepts**.

[ISO 1087-1:2000, 3.2.4]

**3.3.7**
**common attribute**
<metadata> **basic attribute** that is applicable to many or all types of metadata item

**3.3.8**
**conceptual data model**
**data model** that represents an abstract view of the real world

**3.3.9**
**conditional**
required under certain specified conditions

NOTE 1 One of three obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. See also **mandatory** (3.3.18) and **optional** (3.3.29).

NOTE 2 Obligation statuses apply to metadata items with a Registration Status of "recorded" or higher.

**3.3.10**
**coordinate**
measurement from the origin of a frame of reference

**3.3.11**
**data**
re-interpretable representation of information in a formalized manner suitable for communication, interpretation or processing

NOTE Data can be processed by human or automatic means.

[ISO/IEC 2382-1:1993, 01.01.02]

**3.3.12**
**data model**
graphical and/or lexical representation of **data**, specifying their properties, structure and inter-relationships

**3.3.13**
**definition**
representation of a **concept** by a descriptive statement which serves to differentiate it from related concepts

[ISO 1087-1:2000, 3.3.1]

**3.3.14**
**designation**
representation of a **concept** by a sign which denotes it

[ISO 1087-1:2000, 3.4.1]

**3.3.15**
**entity**
any concrete or abstract thing that exists, did exist, or might exist, including associations among these things

EXAMPLE A person, object, event, idea, process, etc…

NOTE Please observe that an entity exists whether data about it are available or not.

[ISO/IEC 2382-17:1999, 17.02.05]

**3.3.16**
**extension**
<11179-3> a feature not defined by ISO/IEC 11179-3

<registry metamodel> a **class**, an **attribute** or a **relationship** that an implementation of a **Metadata Registry** provides that is not defined by ISO/IEC 11179-3

**3.3.17**
**language**
system of signs for communication, usually consisting of a vocabulary and rules

[ISO 5127:2001, 1.1.2.01]

**3.3.18**
**mandatory**
always required

NOTE 1 One of three obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. See also **conditional** (3.3.9) and **optional** (3.3.29).

NOTE 2 Obligation statuses apply to **metadata items** with a **registration status** of "recorded" or higher.

**3.3.19**
**metadata**
**data** that defines and describes other **data**

**3.3.20**
**metadata item**
instance of a **metadata object**

NOTE 1 In all parts of ISO/IEC 11179, this term is applied only to instances of metadata objects described by the metamodel in Clauses 5 through 10 of ISO/IEC 11179-3. Examples include instances of Data_Elements, Data_Element_Concepts, Permissible_Values etc.

NOTE 2 A metadata item has associated attributes, as appropriate for the metadata object it instantiates.

**3.3.21**
**metadata object**
object type defined by a metamodel

NOTE In all parts of ISO/IEC 11179, this term is applied only to metadata objects described by the metamodel in Clauses 5 through 10 of ISO/IEC 11179-3. Examples include Data_Elements, Data_Element_Concepts, Permissible_Values etc.

**3.3.22**
**metadata register**
information store or database maintained by a **Metadata Registry**

**3.3.23**
**Metadata Registry**
**MDR**
information system for registering **metadata**

NOTE The associated information store or database is known as a **metadata register**.

**3.3.24**
**metadata set**
any collection of **metadata**

**3.3.25**
**metamodel**
**model** that specifies one or more other models

**3.3.26**
**metamodel construct**
unit of notation for modelling

NOTE The metamodel constructs used in ISO/IEC 11179-3 are defined in 3.2.

**3.3.27**
**name**
**designation** of an **object** by a linguistic expression

NOTE See also **name** ( attribute of *Individual*) (5.1.5.2.1) and **name** (attribute of *Organization*) (5.1.10.2.1)

**3.3.28**
**object**
anything perceivable or conceivable

NOTE Objects may also be material (e.g. an engine, a sheet of paper, a diamond), immaterial (e.g. a conversion ratio, a project plan) or imagined (e.g. a unicorn).

[Adapted from ISO 1087-1:2000, 3.1.1]

**3.3.29**
**optional**
permitted but not required

NOTE 1 One of three obligation statuses applied to the attributes of metadata items, indicating the conditions under which the attribute is required. See also **conditional** (3.3.9) and **mandatory** (3.3.18).

NOTE 2 Obligation statuses apply to metadata items with a Registration Status of "recorded" or higher.

**3.3.30**
**organization part**
any department, service or other entity within an organization which needs to be identified for information exchange

[ISO/IEC 6523-1:1998, 3.2]

**3.3.31**
**quantity**
Value with an associated unit of measure.

NOTE: 32º Fahrenheit and 0º Celsius are quantities, and they are equivalent values in different measuring systems.

**3.3.32**
**registration**
<generic>inclusion of an item in a registry

<Metadata Registry> inclusion of a **metadata item** in a **Metadata Registry**

**3.3.33**
**registry instance**
implementation of a **registry product.** instance of a **Metadata Registry**

**3.3.34**
**registry item**
**metadata item** recorded in a **Metadata Registry**

**3.3.35**
**registry metamodel**
**metamodel** specifying the model for a **Metadata Registry**

**3.3.36**
**registry product**
**information system** for implementing a **Metadata Registry**

**3.3.37**
**related metadata reference**
**reference** from one **metadata item** to another

NOTE A **Registration_Authority** could choose to use a **Reference_Document**, an **administrative_note** or an **explanatory_comment** to record a **related metadata reference**.

**3.3.38**
**stewardship**
<metadata> the responsibility for the maintenance of administrative information applicable to one or more **administered item**s

NOTE 1 The responsibility for the registration of metadata may be different from the responsibility for stewardship of metadata.

NOTE 2 See also **steward** (3.4.85).

**3.3.39**
**text**
paragraph, page or document that can be used to define or describe an entity. Text is data in the form of characters, symbols, words, phrases, paragraphs, sentences, tables, or other character arrangements, intended to convey a meaning, and whose interpretation is essentially based upon the reader's knowledge of some natural language or artificial language [ISO/IEC 2382-23:1994]

NOTE See also the Text datatype (5.1.17).

## 3.4 Terms used in the specification of the metamodel

This subclause provides definitions for terms which are concepts used in the specification of the metadata model in Clauses 5 through 10. Data definitions are included in those clauses. An alphabetical list of terms with links to the corresponding definitions is included in Annex A.

**3.4.1**
**administered item**
**registered item** for which administrative information is recorded.

**3.4.2**
**antisymmetric relation**
**asymmetric relation** that contains no pair of members in which the same **concept** appears at both ends (in opposite **roles**)

**3.4.3**
**assertion**
sentence or proposition in logic which is asserted (or assumed) to be true.

**3.4.4**
**asymmetric relation**
**binary relation** in which a **role distinction** is made

NOTE 1 In common algebraic notation, this is to say that aRb and bRa are not equivalent.

NOTE *2* Examples of asymmetric relations include: *less than*, *likes*, *father of*, etc.

**3.4.5**
**attached item**
**registered item** for which administrative information is recorded in another **registered item**.

NOTE This is often a member of a group of **registered items** that is managed as a whole.

**3.4.6**
**binary relation**
**relation** of arity 2 (i.e., whose members all have two ends).

NOTE Most common semantic relations are binary, e.g. *equals*, *less than*, *greater than*, *is part of*, etc. An example of a relation which is not binary is *betweenness*.

**3.4.7**
**boolean**
mathematical datatype associated with two-valued logic

**3.4.8**
**characteristic**
abstraction of a property of an **object** or set of objects.

[ISO 1087-1:2000, 3.2.4]

NOTE Characteristics are used for describing concepts.

**3.4.9**
**classifiable item**
**metadata item** of a type for which classification is supported in a given registry.

**3.4.10**
**classification scheme**
descriptive information for an arrangement or division of **objects** into groups based on criteria such as **characteristics**, which the objects have in common

EXAMPLE Origin, composition, structure, application, function, etc.;

**3.4.11**
**concept**
unit of knowledge created by a unique combination of **characteristics**

[ISO 1087-1:2000, 3.2.1]

**3.4.12**
**concept system**
set of **concepts** structured according to the **relations** among them. [ISO 1087-1]

**3.4.13**
**conceptual domain**
**CD**
**concept** that expresses its valid instance meanings or description

**3.4.14**
**contact**
instance of a role of an individual or an organization (or organization part or organization person) to whom an information item(s), a material object(s) and/or person(s) can be sent to or from in a specified context

**3.4.15**
**contact individual**
**individual** that is the **contact**

**3.4.16**
**contact organization**
**organization** for which the **contact** acts as a representative

**3.4.17**
**context**
<designation and definition> universe of discourse in which a **designation** or **definition** is used

**3.4.18**
**data element**
**DE**

EDITOR'S NOTE #4. (Action required) There has been feedback from users of 11179 that this is not a useful definition. It should be reviewed.

unit of **data** for which the **definition**, **identification**, representation and **value domain** are specified by means of a set of **attribute**s

**3.4.19**
**data element concept**
**DEC**
specification of a **concept** independent of any particular representation

NOTE A **data element** is a representation of a **data element concept**.

**3.4.20**
**data element concept characteristic**
**characteristic** of a **data element concept**

**3.4.21**
**data element concept domain**
**conceptual domain** of a **data element concept**

**3.4.22**
**data element concept object class**
**object class** of a **data element concept**

**3.4.23**
**data element derivation**
**association** among a **data element** which is derived, the **derivation rule** controlling its derivation, and the **data element**(s) from which it is derived

**3.4.24**
**data element domain**
**value domain** of a **data element**

**3.4.25**
**data element example**
representative illustration of a **data element**

**3.4.26**
**data element meaning**
**data element concept** that provides meaning for a **data element**.

**3.4.27**
**data element precision**
degree of specificity for a **data element**

NOTE Expressed as a number of decimal places to be used in any associated **data element** values.

**3.4.28**
**date and time**
family of **datatypes** whose values are points in time to various common resolutions: year, month, day, hour, minute, second, and fractions thereof. (see also 5.1.4)

Based on [ISO/IEC 11404:2007, 8.1.6 time]

**3.4.29**
**definition**
<designatable item> representation of a **designatable item** by a descriptive statement which, in a given **language** and **context**(s) serves to differentiate it from related designatable items

NOTE See also **definition** (3.3.13 above).

**3.4.30**
**definition context**
<designatable item> **context** in which the **definition** is applicable

**3.4.31**
**definition language**
**language** used to write the **definition_text**

**3.4.32**
**definition source reference**
reference to the source from which the **definition** is taken

**3.4.33**
**definition text**
text of the **definition**

**3.4.34**
**derivation rule**
logical, mathematical, and/or other operations specifying derivation

**3.4.35**
**derivation rule notation**
**notation** used to describe the **derivation rule**

**3.4.36**
**derivation rule specification**
text of a specification of **data element derivation**

**3.4.37**
**described conceptual domain**
**conceptual domain** that is specified by a description

**3.4.38**
**described value domain**
**value domain** that is specified by a description

**3.4.39**
**designatable item**
**Identified item** which can have **designations** and/or **definitions.**

**3.4.40**
**designation**
<designatable item> representation of a **designatable item** by a **sign** which denotes it.

NOTE See also **designation** (3.?.??).

**3.4.41**
**designation acceptability**
rating of the acceptability of the **designation** in the specified **context**

**3.4.42**
**designation context**
<designatable item> **context** in which a **designation** is applicable

**3.4.43**
**designation language**
<designatable item> **language** or dialect in which a **sign** (usually a **name**) is expressed.

**3.4.44**
**designation namespace**
**namespace** to which a **designation** is bound

**3.4.45**
**designation sign**
<designatable item> **sign** of the **designation**.

**3.4.46**
**dimensionality**
set of equivalent units of measure, where equivalence between two units of measure is determined by the existence of a quantity preserving one-to-one correspondence between values measured in one unit of measure and values measured in the other unit of measure, independent of context, and where characterizing operations are the same.

NOTE 1 The equivalence defined here forms an equivalence relation on the set of all units of measure. Each equivalence class corresponds to a dimensionality. The units of measure "temperature in degrees Fahrenheit" and "temperature in degrees Celsius" have the same dimensionality, because:
a) given a value measured in degrees Fahrenheit there is a value measured in degrees Celsius with the same quantity, and vice-versa, by the well-known correspondences $C^o = (5/9)*(F^o - 32)$ and $F^o = (9/5)*(C^o) + 32$.
b) the same operations can be performed on both values.

NOTE 2 The units of measure "temperature in degrees Celsius" and "temperature in degrees Kelvin" do not belong to the same dimensionality. Even though it is easy to convert quantities from one unit of measure to the other ($C^o = K^o - 273.15$ and $K^o = C^o + 273.15$), the characterizing operations in degrees Kelvin include taking ratios, whereas this is not the case for degrees Celsius. For instance, $20^o$ K is twice as warm as $10^o$ K, but $20^o$ C is not twice as warm as $10^o$ C.

NOTE 3 Units of measure are not limited to physical categories. Examples of physical categories are: linear measure, area, volume, mass, velocity, time duration. Examples of non-physical categories are: currency, quality indicator, colour intensity

NOTE 4 Quantities may be grouped together into categories of quantities which are mutually comparable. Lengths, diameters, distances, heights, wavelengths and so on would constitute such a category. Mutually comparable quantities have the same dimensionality. ISO 31-0 calls these “quantities of the same kind”.

NOTE 5 ISO 31-0 specifies physical dimensions (e.g. length, mass, velocity). ISO/IEC 11179-3 also permits non-physical dimensions (e.g. value dimensions such as: currency, quality indicator). The present concept of dimensionality equates to what ISO 31 calls Dimensional Product, rather than to Dimension.

NOTE 6 See also **unit of measure** (3.4.94).

**3.4.47**
**enumerated conceptual domain**
**conceptual domain** that is specified by a list of all its **value meaning**s

NOTE No ordering of the value meanings is implied

**3.4.48**
**enumerated value domain**
**value domain** that is specified by a list of all its **permissible value**s

NOTE No ordering of the permissible values is implied

**3.4.49**
**identified item**
**metadata item** identified in a **metadata registry**

**3.4.50**
**individual**
single human being.

**3.4.51**
**integer**
mathematical datatype comprising the exact integral values

[ISO/IEC 11404:2007, 8.1.7]

**3.4.52**
**international code designator**
**identifier** of an organization identification scheme

NOTE 1 Based on ISO/IEC 6523-1:1998, 3.8.

NOTE 2 See also ISO/IEC 11179-6.

**3.4.53**
**management**
<metadata registry>**association** denoting the **registration authority** that is responsible for managing and maintaining the **register**.

**3.4.54**
**namespace**
set of **designations** for a particular business need.

**3.4.55**
**naming convention**
specification of how **signs** of **designations** are formulated.

**3.4.56**
**notation**
formal syntax and semantics, meant for machine processing.

Examples: UML, MOF, OCL, OWL/RDF, SKOS, CGIF, XCL, XTM, or ISO/IEC 11404

**3.4.57**
**object class**
set of ideas, abstractions or things in the real world that are identified with explicit boundaries and meaning and whose properties and behaviour follow the same rules.

**3.4.58**
**organization**
unique framework of authority within which **individuals** act, or are designated to act, towards some purpose

NOTE The kinds of organizations covered by ISO/IEC 6523-1 include the following examples:

a) an organization incorporated under law;

b) an unincorporated organization or activity providing goods and/or services including:

   1) partnerships;

   2) social or other non-profit organizations or similar bodies in which ownership or control is vested in a group of individuals;

   3) sole proprietorships

   4) governmental bodies .

c) groupings of the above types of organizations where there is a need to identify these in information interchange.

[Adapted from ISO/IEC 6523-1:1998, 3.1]

**3.4.59**
**organization identifier**
**identifier** assigned to an **organization** within an organization identification scheme, and unique within that scheme

[ISO/IEC 6523-1:1998, 3.10]

**3.4.60**
**organization part identifier**
**opi**
**identifier** allocated to a particular **organization part**

NOTE See also ISO/IEC 11179-6.

[ISO/IEC 6523-1:1998, 3.11]

**3.4.61**
**permissible value**
**designation** of a **value meaning** within a specific **value domain**

**3.4.62**
**phone number**
telephone number

NOTE Specified by ITU-T Recommendation E.164 (2005-02), the international public telecommunications numbering plan.

**3.4.63**
**postal address**
set of information which, for a postal item, allows the unambiguous determination of an actual or potential delivery point, usually combined with the specification of an addressee and/or a mailee.

[UPU S42]

**3.4.64**
**reference document**
document that provides pertinent details for consultation about a subject

**3.4.65**
**reference document identifier**
**identifier** for the **reference document**

**3.4.66**
**reference document language identifier**
**identifier** of the natural or special **language** used in the **reference document**

**3.4.67**
**reference document notation**
notation used within the reference document

**3.4.68**
**reference document title**
title of the **reference document**

**3.4.69**
**reference document type description**
description of the type of **reference document**

**3.4.70**
**reference document uri**
universal resource identifier (uri) for the **reference document**

**3.4.71**
**reference provider**

EDITOR'S NOTE #5. (Action Required) Do we need to be able to distinguish different types of reference_provider? For example, one organization might maintain the document, but another might publish it or make it available.

**organization** that maintains or carries an official copy of the reference document.

**3.4.72**
**reflexive relation**
**binary relation** in which all elements of the set are related to themselves

**3.4.73**
**register**
data store where **registered items** are recorded and managed.

**3.4.74**
**registered Item**
**metadata item** that is recorded and managed in a **metadata registry**.

**3.4.75**
**registrar**
representative of a **registration authority**

**3.4.76**
**registrar identifier**
**identifier** for the **registrar**

**3.4.77**
**registration authority**
**RA**
**organization** responsible for maintaining a **register**

**3.4.78**
**registration authority identifier**
**identifier** assigned to a **registration authority**

NOTE See ISO/IEC 11179-6 and ISO/IEC 6523-2.

**3.4.79**
**registration record**
<registration> information about the **registration** of an **administered item**

**3.4.80**
**registration status**
**designation** of the status in the registration life-cycle of an **administered item**

NOTE Designation values are described in ISO/IEC 11179-6.

**3.4.81**
**relation**
subset of the powerset of RxUD, for some role set R, where UD is the universe of discourse

NOTE 1 An *n*-ary relation on sets $A_1$, ..., $A_n$ is a subset of Cartesian product $A_1$ x ... x $A_n$ .

NOTE 2 Membership of an n-tuple in the relation is specified by means of a predicate which must be true for the n-tuple to be a member of the corresponding relation.

NOTE 3 In the 11179-3 metamodel, **relations** are defined over sets of **concepts**.

**3.4.82**
**scoped identifier**
**identifier** of an **identified item** within a specified **namespace**

NOTE A **namespace** provides the scope within which the **scoped identifier** uniquely identifies the **identified item**.

**3.4.83**
**sign** (noun)
textual string or symbol that can be used to denote a concept

**3.4.84**
**slot**
container for extensions to **identified items**

**3.4.85**
**steward**
**organization** that maintains **stewardship** of an **administered item**

**3.4.86**
**stewardship contact**
**contact** information associated with a **stewardship**

**3.4.87**
**stewardship record**
record of a **steward** (an **organization**) and a **stewardship contact** (a **contact**) involved in the **stewardship** of an **administered item**

**3.4.88**
**string**
family of datatypes which represent strings of symbols from standard character-sets.

[ISO/IEC 11404:2007 10.1.5 Character String]

NOTE The syntax and semantics of the String datatype are as defined in ISO/IEC 11404:2007 10.1.5 Character String

**3.4.89**
**submission**
**act** of submitting a **metadata item** for inclusion in a **metadata registry**

EDITOR'S NOTE #6. (Information) The editor has changed the above definition, because the previous definition was defined as an association with *Submission_Record,* but that is defined in terms of *submission*, so the result was circular.

**3.4.90**
**submission contact**
**contact** information associated with a **submission**

**3.4.91**
**submission record**
record of a **submitter** (an **organization**) and a **submission contact** (a **contact**) involved in the **submission** of a **metadata item** for **registration**

EDITOR'S NOTE #7. (Information) The editor has changed the above definition, replacing **registered item** by **metadata item** for **registration**, because the item is not yet registered when it is submitted.

**3.4.92**
**symmetric relation**
**binary relation** where a **link** between any two **concepts,** Concept A and Concept B, necessarily means that a **link** of the same type also exists in the opposite direction between Concept B and Concept A

**3.4.93**
**transitive relation**
a **binary relation** *R* over a set *X* is transitive if whenever an element *a* is related to an element *b*, and *b* is in turn related to an element *c*, then *a* is also related to *c*.

**3.4.94**
**unit of measure**
⟨value domain⟩ actual units in which the associated values are measured

NOTE 1 ISO 31-0:1982 specifies a system of physical measurement (the International System of Units, SI). Physical measurement is only one type of measurement. Value measurement is another type of measurement. ISO/IEC 11179-3 permits the use of any appropriate system of measurement.

NOTE 2 The **dimensionality** of the associated **conceptual domain** must be appropriate for the specified **unit of measure**.

**3.4.95**
**unit of measure dimensionality**
**dimensionality** that specifies the equivalence relation that applies to all values representing this particular unit.

**3.4.96**
**value domain**
**VD**
set of **permissible value**s

NOTE 1 The **value domain** provides representation, but has no implication as to what **data element concept** the **value**s may be associated with nor what the **Value**s mean

NOTE 2 The **permissible value**s may either be enumerated or expressed via a description.

**3.4.97**
**value meaning**
semantic content of a **value**

NOTE 2 The representation of **Value_Meaning**s in a registry shall be independent of (and shall not constrain) their representation in any corresponding **Value_Domain**.

**3.4.98**
**version**
unique version **identifier** of the **administered item**

## 3.5 List of Abbreviations and Acronyms

The following abbreviations and acronyms are defined for use within the subject domain of this document.

**3.5.1**
**CD**
Conceptual Domain

**3.5.2**
**DE**
Data Element

**3.5.3**
**DEC**
Data Element Concept

**3.5.4**
**MDR**
Metadata Registry

**3.5.5**
**opi**
organization_part_identifier

**3.5.6**
**ORM**
Object Role Modelling

**3.5.7**
**OWL**
Web Ontology Language

**3.5.8**
**OWL-DL**
OWL Description Logic

**3.5.9**
**RA**
Registration Authority

**3.5.10**
**RDF**
Resource Description Framework

**3.5.11**
**SKOS**
Simple Knowledge Organization System

**3.5.12**
**UML**
Unified Modeling Language

**3.5.13**
**UPU**
Universal Postal Union

**3.5.14**
**VD**
Value Domain

**3.5.15**
**W3C**
World Wide Web Consortium

**3.5.16**
**XCL**
eXtended Common Logic markup language

**3.5.17**
**XML**
eXtensible Markup Language

**3.5.18**
**XTML**
eXtensible Topic Maps

# 4 Structure of a Metadata Registry

## 4.1 Metamodel for a Metadata Registry

A metamodel is a model that describes other models. A metamodel provides a mechanism for understanding the precise structure and components of the specified models, which are needed for the successful sharing of the models by users and/or software facilities.

This part of ISO/IEC 11179 uses a metamodel to describe the structure of a *Metadata Registry*. The registry in turn will be used to describe and model other data, for example about enterprise, public administration or business applications. The *registry metamodel* is specified as a conceptual data model, i.e. one that describes how relevant information is structured in the natural world. In other words, it is how the human mind is accustomed to thinking of the information.

As a conceptual data model, there need be no one-to-one match between the attributes in the model and fields, columns, objects, et cetera in a database. There may be more than one field per attribute and some entities and relationships may be implemented as fields. There is no intent that an implementation should have a table for each relationship or entity. The metamodel need not be physically implemented as specified.

The structure described by this metamodel may be distributed over several implementations. These implementations may be databases, data repositories, metadata registers, metadata registries, dictionaries, etc. No particular technology is implied. Implementations may utilize technologies including, but not limited to: relational database, XML database, object oriented systems, or RDF/OWL.

The model shows constraints on minimum and maximum occurrences (the obligation) of attributes. The constraints on maximum occurrences are to be enforced at all times. The constraints on minimum occurrences are to be enforced when the registration_status for the metadata item is "recorded" or higher. In other words, a registration_status of "recorded" indicates that all mandatory attributes have been documented.

## 4.2 Application of the metamodel

Some of the objectives of the metamodel for a *Metadata Registry* are to:

— provide a unified view of concepts, terms, value domains and value meanings;

— promote a common understanding of the data described;

— provide the specification at a conceptual level to facilitate the sharing and reuse of the contents of implementations.

A metamodel is necessary for coordination of data representation between persons and/or systems that store, manipulate and exchange data. The metamodel will assist registrars in maintaining consistency among different registries. The metamodel enables systems tools and information registries to store, manipulate and exchange the metadata for data attribution, classification, definition, naming, identification, and registration. In this manner, consistency of data content supports interoperability among systems tools and information registries.

Using the metamodel, mappings to the schema of particular metadata management tool sets can be developed. The metamodel constructs can be translated into the language of each tool set, preserving the concepts represented in the metamodel.

An implementer may use this conceptual data model to develop a more specific logical data model of the identical sphere of interest. A logical data model describes the same data, but as structured in an information system. It is often referred to as a Model of the Information System. A logical data model can be directly used for database design.

## 4.3 Specification of the metamodel

### 4.3.1 Terminology used in specifying the metamodel

When using a model to specify another model, it is easy for the reader to become confused about which model is being referred to at any particular point. To minimize this confusion, this document deliberately uses different terms in the model being specified from those used to do the specification.

The *registry metamodel* is specified using a subset of the Unified Modelling Language (UML) Version 2.1.2. This document uses the term "metamodel construct" for the UML model constructs it uses, but "metadata objects" for the model constructs it specifies. The metamodel constructs used are: classes, associations, association classes, attributes, composite attributes and composite datatypes, and these are defined in 3.2. The specified metadata objects are defined in clauses 5 through 10. The concepts that the metadata objects represent are defined in clause 3.4.

However, there are certain parallels between the two metamodels. For example, the "Object_Class" specified in the registry metamodel is similar to the metamodel construct “Class”, and the “Characteristic” specified in the registry metamodel is similar to the metamodel construct “Attribute”. The different terms are used to make it clear which model is being referred to, not because they represent different concepts. One term that this document uses at both levels is “datatype”, but the level to which it applies should be apparent from the context in which it is used.

### 4.3.2 Use of UML Packages

For descriptive and conformance purposes, the metamodel is organized into packages:

- **Basic package** (clause 5) – contains simple classes that are reused by other packages
- **Identification, designation and definition package** (clause 6) – contains classes that enable the contents of a registry to be identified, named or otherwise designated, and defined. This package is sub-divided into two regions:
  - Identification region (see 6.1)
  - Designation and Definition region (see 6.2)
- **Registration package** (clause 7) – contains classes that enable metadata items to be registered
- **Concepts package** (clause 8) – contains classes that enable concepts to be related. This package is sub-divided into two regions:
  - Concepts System region (see 8.1)
  - Classification region (see 8.2)
- **Binary Relations Package** (clause 9) – contains a specialization of the Relations class from the Concepts package, specifically for binary relations. The separation is for conformance purposes.
- **Data Descriptions Package** (clause 10) – contains classes that enable the description of specific metadata objects:
  - Data Element Concepts region (see 10.2)
  - Conceptual and Value_Domains region (see 10.3)
  - Data_Elements region (see 10.4).

### 4.3.3 Package Dependencies

Figure 4-1 illustrates the dependencies among the packages.

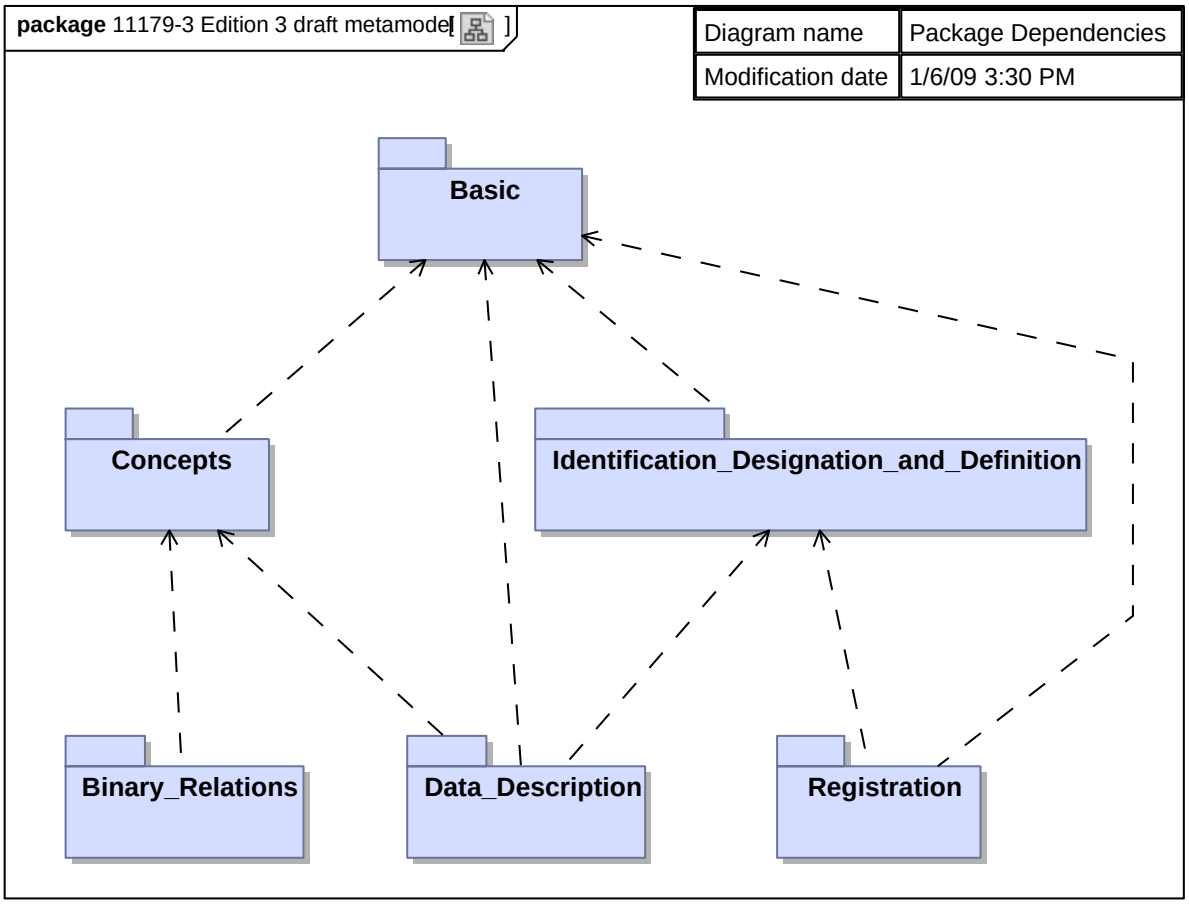



**Figure 4-1 — Package dependencies**

### 4.3.4 Use of UML Class diagrams and textual description

This standard uses both text and UML class diagrams to describe the metamodel. Both are normative, and are intended to be complementary. However, if a conflict exists between what is specified in UML and what is specified in text, the text takes precedence until such time as a correction is made to make them consistent.

A consolidated UML class hierarchy is included as Annex B.

## 4.4 Types, Instances and Values

When considering data and metadata, it is important to distinguish between types of data/metadata, and instances of these types and their associated values. The metamodel specifies types of classes, attributes and associations. Any particular instance of one of these will be of a specific type, and at any point in time, that instance will have a specific value (possibly null). As examples, this document defines *attribute instance* and *attribute value*, but the same principle applies to classes, relationships and all other metamodel constructs defined in 3.2.

NOTE In UML, sub-classes of a super-class are by default **not** disjoint. This standard specifies when sub-classes are required to be disjoint. Further, when the list of sub-classes is intended to be exhaustive, this standard shows the

super-class as abstract, thus preventing any other sub-class being instantiated. An abstract class is indicated by showing the class name in *italics* in any class diagram that uses it.

Clauses 5 through 10 of this document specify the types of *metadata objects* that form the structure of a *metadata registry*. A *metadata registry* will be populated with instances of these *metadata objects* (referred to as *metadata items*), which in turn define types of data, e.g. in an application database. In other words, instances of metadata specify types of application level data. In turn, the application database will be populated by the real world data as instances of those defined metadata object types.

NOTE ISO/IEC TR 10032:2003 Reference model for data management explains the concepts of different levels of modelling, as does ISO/IEC 10027:1990 IRDS Framework.

### 4.5 Types of Items in an ISO/IEC 11179 metadata registry

Figure 4-2 shows the types of items specified by this Part of this International Standard. These types are explained in subsequent clauses.

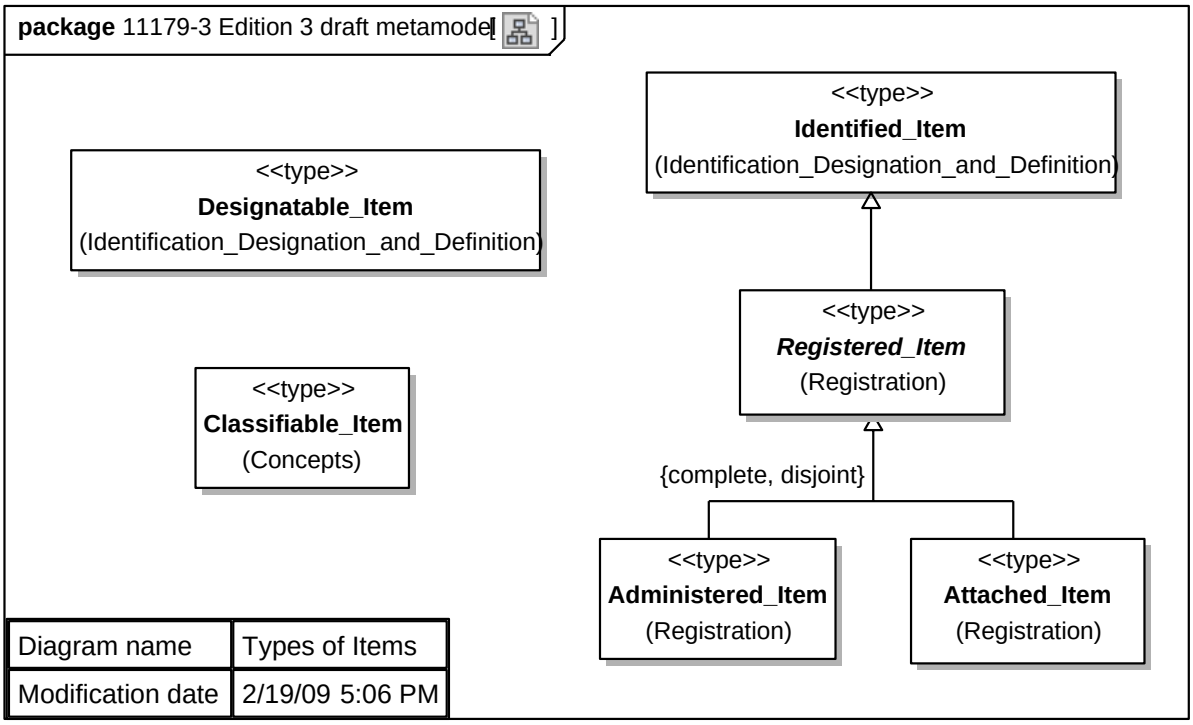



**Figure 4-2 — Types of items**

Any *metadata item* entered into a *metadata registry* may be extended by one or more of the above types, as follows:

— Any *metadata item* that is to be retrieved directly (as opposed to indirectly through a related item), shall be an *Identified_Item* (see 6.1.2.1), so the item may be referenced. An example of metadata items that might not be explicitly identified are the *permissible values* within a *value domain*.

— Any *metadata item* that is to be designated (named) and/or defined shall be a *Designatable_Item* (see 6.2.2.1).

Note: The separation of Designation and Definition from Identification has been done to better harmonize with ISO/IEC 19763-2, in which ModelElements are identified and are administered, but are not (required to be) designated or defined.

— Any *metadata item* that is to be registered in the registry shall be a *Registered_Item* (see 7.1.2.1). *Registered_Item* is an abstract class, which means that each such item must be instantiated as one of the subtypes: *Administered_Item* (see 0), or *Attached_Item* (see 7.1.2.3). These subtypes are exhaustive and mutually exclusive.

— Any *metadata item* that is to be classified in a *classification scheme* shall be a *Classifiable_Item* (see 8.2.2.1).

A *Registration_Authority* responsible for the registry shall determine which *metadata items* should become *Identified_Items*, *Registered_Items*, *Designatable_Items* and/or *Classifiable_Items*, within the contraints of any conformance claim that is made for the registry. (See 12 Conformance.)

### 4.5.1 Rules for Identified_Item and its subtypes.

The following table is a compressed decision table that shows the rules that apply to the various types of item.

| **Item instance has instance of:** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** | **Rule 5** | **Rule 6** | **Rule 7** |
|---|---|---|---|---|---|---|---|
| Scoped_Identifier | one or more | | | | | | |
| Designation | | one or more | | | | | |
| Definition | | | one or more | | | | |
| Classification | | | | one or more | | | |
| Submission_Record | | | | | one or more | | |
| Stewardship_Record | | | | | | one | none |
| Registration | | | | | | one or more | none |
| Attachment | | | | | | | one or more |
| Item is identified | X | | | | | | |
| Item is designated | | X | | | | | |
| Item is defined | | | X | | | | |
| Item is classified | | | | X | | | |
| Item is registered | | | | | X | | |
| Item is administered | | | | | | X | |
| Item is attached | | | | | | | X |

**Table 4-1: Rules for Types of Items**

## 4.6 Extensibility

It is not expected that this metamodel will completely accommodate all users. Particular sectors, such as document management, scientific data, statistical data, require metadata attributes not addressed in this standard. This standard provides *Slots* (see 6.1.2.4) as a mechanism to extend metadata items with custom attributes. Classes, relationships, and attributes may be added as extensions to existing packages in this conceptual data model, or complete new packages may be added.

Implementers of this standard may include extensions as part of an implementation, and/or they may provide facilities to enable a registry user to define their own extensions, such as classes and/or packages. An implementation with such extensions shall be considered conformant if it does not violate any of the rules inherent in the structure and content as specified by the metamodel in this standard.

## 4.7 Date References

In this standard, dates are important attributes of an *Administered_Item* and of operations of a registry. For the purpose of this standard, “date” refers to Gregorian calendar date {see ISO 8601:2000}. (See also 5.1.4 Date-and-time for the specification of the associated datatype.)

# 5 Basic Package

## 5.1 Basic types and classes metamodel region

### 5.1.1 Overview

The Basic package specifies common datatypes for use elsewhere in the metamodel. A datatype is a set of distinct values, characterized by properties of those values and by operations on those values (ISO/IEC 11404). All of the other types used in the model are based on this core set of types, and any compliant implementation of a metadata registry should include an implementation of the semantics specified in these core types.

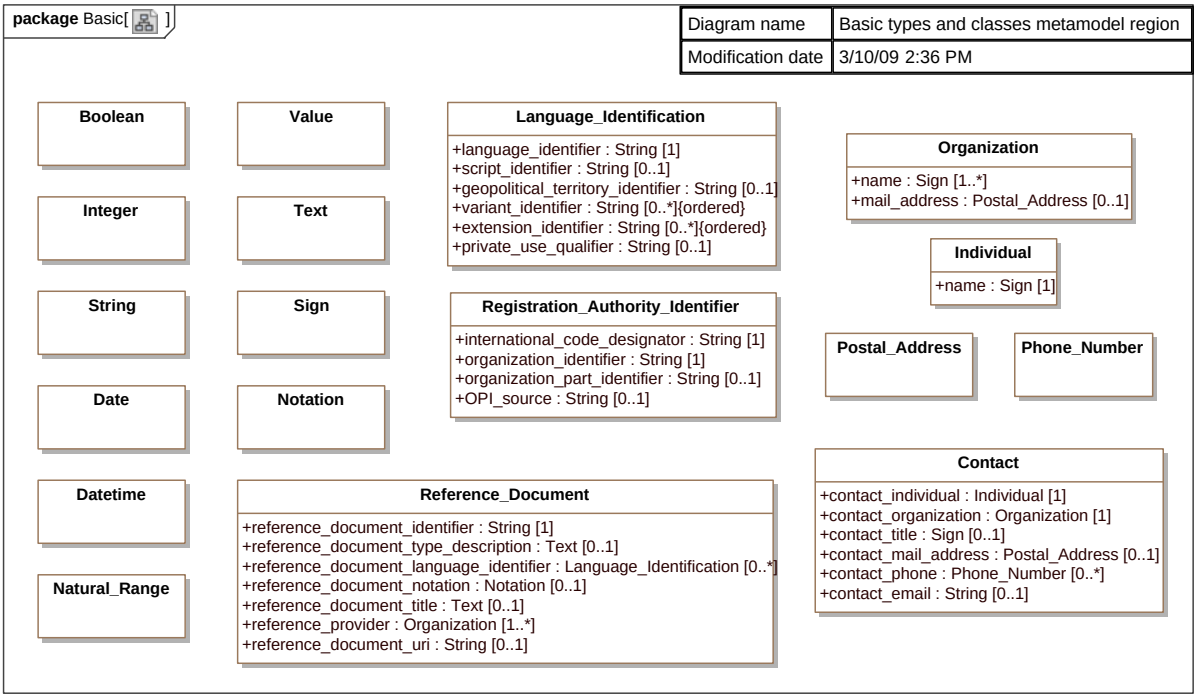



**Figure 5-1 — Basic types and classes metamodel region**

### 5.1.2 Boolean

A mathematical datatype associated with two-valued logic. [ISO/IEC 11404:2007, 8.1.1 Boolean].

NOTE The notation and semantics for Boolean is as described in ISO/IEC 11404.

### 5.1.3 Contact

#### 5.1.3.1 Description of *Contact*

*Contact* is the class of object to whom information item(s), material object(s) and/or person(s) can be sent to or from. *Registrar* (see 7.1.2.4) is a subtype of *Contact*.

The attributes of the *Contact* class are summarized here and specified more formally in 5.1.3.2.

— Every *Contact* shall have exactly one *contact_organization* of type *Organization* for which the *Contact* is a representative.

— Every *Contact* shall have exactly one *contact_individual* of type *Individual.* The *contact_individual* is the *Individual* that the *Contact* information relates to.

— A *Contact* may have zero or one *contact_titles* of type *Sign*, that identifies the position held by the *contact_individual*.

— A *Contact* may have zero or one *contact_mail_addresses* of type *Postal_Address*, where the *contact_individual* may be contacted by postal mail.

— A *Contact* may have zero or more *contact_phones* of type *Phone_Number* where the *contact_individual* may be contacted by phone.

— A *Contact* may have zero or one *contact_email* addresses of type *String*, where the *contact_individual* may be contacted by e-mail.

#### 5.1.3.2 Attributes of *Contact*

##### 5.1.3.2.1 contact_individual

Attribute name: ***contact_individual***

Definition: *Individual* that is the *Contact*

Obligation: Mandatory

Multiplicity: *1*

Data type: *Individual* (5.1.5)

##### 5.1.3.2.2 contact_organization

Attribute name: ***contact_organization***

Definition: *Organization* for which the *Contact* acts as a representative

Obligation: Mandatory

Multiplicity: *1*

Data type: *Organization* (5.1.10)

##### 5.1.3.2.3 contact_title

Attribute name: ***contact_title***

Definition: name of the position held by the *Contact*

Obligation: Optional

Multiplicity: 0..1

Data type: *Sign* (5.1.15)

##### 5.1.3.2.4 contact_mail_address

Attribute name: ***contact_mail_address***

Definition: postal address for the *Contact*

Obligation: Optional

Multiplicity: 0..1

Data type: *Postal_Address* (5.1.12)

##### 5.1.3.2.5 contact_phone

Attribute name: ***contact_phone***

Definition: phone number for the *Contact*

Obligation: Optional

Multiplicity: 0..*

Data type: *Phone_Number* (5.1.11)

#### 5.1.3.2.6 contact_email

Attribute name: ***contact_email***

Definition: email address of the *Contact.*

Obligation: Optional

Multiplicity: 0..1

Data type: *String* (5.1.16)

—— End of attributes of *Contact* ——

### 5.1.4 Date-and-time

A family of datatypes whose values are points in time to various common resolutions: year, month, day, hour, minute, second, and fractions thereof [ISO/IEC 11404:2007, 8.1.6 Date-and-Time].

In this standard we use:

— Date – which includes year, month and day

— Datetime – which includes year, month, day, hours, minutes and seconds, and optionally fractions of seconds.

### 5.1.5 Individual

#### 5.1.5.1 Description of *Individual*

*Individual* is a class whose instances represent a single human being. The *Individual* class has one attribute:

— Every *Individual* shall have exactly one *name* of type *Sign*.

#### 5.1.5.2 Attributes of *Individual*

#### 5.1.5.2.1 name

Attribute name: ***name***

Definition: *Sign* that designates the *Individual*.

Obligation: Mandatory

Multiplicity: *1*

Data type: *Sign* (5.1.15)

—— End of attributes of *Individual* ——

### 5.1.6 Integer

A mathematical datatype comprising the exact integral values [ISO/IEC 11404:2007, 8.1.7 Integer].

NOTE Both the notation and semantics of the Integer datatype is as specified in ISO/IEC 11404:2007:8.1.7.

### 5.1.7 Language_Identification

#### 5.1.7.1 Description of *Language_Identification*

The composite datatype *Language_Identification* serves as an identifier for a language. *Language_Identification* always defines a language as spoken (or written, signed or otherwise signaled) by human beings for communication of information to other human beings. Computer languages such as programming languages are explicitly excluded.

The identifier is comprised of the following parts, which are based on IETF RFC 4646:

— a mandatory *language_identifier* that identifies the primary language

— an optional *script_identifier* that identifies the set of graphic characters used for the written form of one or more languages

— an optional *region_identifier* that denotes the area or region in which a word, term, phrase or language variant is used.

— zero or more *variant_identifiers* that denotes a specific variant or variants of a given language. Variant identifiers are typically represented as dates and are used to distinguish events such as spelling reforms.

— zero or more *extension_identifiers* that denote extensions to a given language. Extensions consist of key-value pairs, which may be order dependent.

— an optional *private use qualifier* that provides additional qualification for specific non-standardized purposes and uses.

NOTE 1 The W3C has a description of the use of the IETF language subtags at: http://www.w3.org/International/articles/language-tags/Overview.en.php.

NOTE 2 IANA maintains a registry of language subtags at: http://www.iana.org/assignments/language-subtag-registry.

NOTE 3 RFC 4646 requires the extension identifiers to be prefixed by a single character that identifies the registration authority that has registered the extension.

#### 5.1.7.2 Attributes of *Language_Identification*

##### 5.1.7.2.1 language_identifier

| | |
|---|---|
| Attribute name: | ***language_identifier*** |
| Definition: | *identifier* for the *Language* |
| Obligation: | Mandatory |
| Multiplicity | *1* |
| Data type: | *String* (5.1.16) |
| Note: | Use the three character alphabetic codes from ISO 639-2/Terminology, with extensions if required. |

##### 5.1.7.2.2 script_identifier

| | |
|---|---|
| Attribute name: | ***script_identifier*** |
| Definition: | identifies the set of graphic characters used for the written form of one or more languages |
| Obligation: | Optional |
| Multiplicity | *0..1* |
| Data type: | *String* (5.1.16) |

| | |
|---|---|
| Note: | Use the four character codes from ISO 15924:2004 codes for the representation of the names of scripts. |

#### 5.1.7.2.3 region_identifier

EDITOR'S NOTE #8. (Action required) Several of the changes proposed by Issue 240 are intended to support IETF RFC 4646 Tags for Identifying Languages. However, RFC 4646 recommends using ISO 3166-1 2-char alpha codes where available, and 3 digit numeric codes where no 2-char alpha code exists. 11179-3 edition 2 specified the use of 3 digit numeric codes, with extensions if necessary. In RFC 4646, extensions are supported either through the use of 2-char alpha codes reserved for private use by 3166-1, or by separate extension and private use identifiers. We need to decide which approach to use in edition 3. An application needing 2 char alpha codes, could translate from the 3 digit numeric code.

| | |
|---|---|
| Attribute name: | ***region_identifier*** |
| Definition: | identifies a specific country, territory, or region whose linguistic variations apply |
| Obligation: | Optional |
| Multiplicity | *0..1* |
| Data type: | *String* (5.1.16) |
| Note: | Use the three digit numeric codes from ISO 3166-1, with extensions if required. |

#### 5.1.7.2.4 variant_identifier

| | |
|---|---|
| Attribute name: | ***variant_identifier*** |
| Definition: | identifies a language variant, which indicates additional, well-recognized variations that define a language or its dialects that are not covered by other available identifiers |
| Obligation: | Optional |
| Multiplicity | *0..** |
| Data type: | *String* (5.1.16) [ordered] |
| Note: | Variant identifiers are typically represented as dates and are used distinguish events such as spelling reforms. Variant identifiers can be order dependent. String Numeric variant_identifiers are interpreted to be Gregorian calendar year numbers. Alphanumeric tags reference IANA variant subtags. |

#### 5.1.7.2.5 extension_identifier

| | |
|---|---|
| Attribute name: | ***extension_identifier*** |
| Definition: | identifies an extension to a *language_identifier* |
| Obligation: | Optional |
| Multiplicity | *0..** |
| Data type: | *String* (5.1.16) [ordered] |
| Note 1: | Extension identifiers are ordered and consist of key-value pairs, separated by the EQUALS SIGN (=). The values must be alphanumeric with no embedded white-space. Whitespace separates the identifiers. |
| Note 2: | As of 2009-03-23, no extensions have been registered. |

#### 5.1.7.2.6 private_use_qualifer

| | |
|---|---|
| Attribute name: | ***private_use_qualifer*** |
| Definition: | qualifier whose meaning is defined solely by private agreement. |
| Obligation: | Optional |
| Multiplicity | *0..1* |

Data type: *String* (5.1.16)

Note: Definition derived from IETF RFC 4646.

—— End of attributes of *Language_Identification* ——

### 5.1.8 Natural_Range

*Natural_Range* is a datatype comprising a range of "natural numbers", i.e. the positive integers, including zero. Any instance of *Natural_Range* is one of:

— a constant non-negative Integer

— a bounded range of non-negative Integers defined by a minimum and a (strictly larger) maximum value

— an unbounded range defined by only a non-negative minimum (e.g., 0..*, 1..*, 2..*).

Note: *Natural_Range* is used as the type of both *multiplicity* (an attribute of *Relation Roles*) and *arity* (an attribute of *Relations*) in the Concepts metamodel region.

### 5.1.9 Notation

*Notation* denotes a notation defined elsewhere, but used by an item within the registry. A notation defines a formal syntax and semantics, meant for machine processing. In this metamodel, *Notation* is used by *Concept_System* (8.1.2.2) and *Reference_Document* (5.1.13).

Examples of such notations include XCL Common Logic (ISO/IEC 24707) or OWL-DL XML notation.

### 5.1.10 Organization

#### 5.1.10.1 Description of *Organization*

*Organization* is a class whose instances represent a unique framework of authority within which *individuals* act, or are designated to act, towards some purpose.

The attributes of the *Organization* class are summarized here and specified more formally in 5.1.10.2.

— Every *Organization* shall have one or more *name*s of type *Sign*.

— An *Organization* may have zero or one *mail_address*es of type *Postal_Address,* where the *Organization* can be contacted by postal mail.

#### 5.1.10.2 Attributes of *Organization*

##### 5.1.10.2.1 name

Attribute name: ***name***

Definition: *Sign* that designates the *Organization.*

Obligation: Mandatory

Multiplicity: *1..**

Data type: *Sign* (5.1.15)

##### 5.1.10.2.2 mail address

Attribute name: ***mail_address***

Definition: postal address for the *Organization.*

| | |
|---|---|
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Postal_Address* (5.1.12) |

—— End of attributes of *Organization* ——

### 5.1.11 Phone_Number

EDITOR'S NOTE #9. (Action required) Do we need to reference both ITU-T E.164 and ISO/IEC 19773 module 17. What value does the reference to ISO/IEC 19773 add here?

A phone number uniquely identifies a telephone line within a telephone network. The data structure of the Phone_Number data element shall conform to ITU-T E 164 and may conform to ISO 19773 Information technology – Metadata registries (MDR) Modules – Module 17: Data structure for ITU-T E.164 phone number data.

### 5.1.12 Postal_address

EDITOR'S NOTE #10. (Action required) Should we reference UPU S42 directly in addition to or instead of ISO/IEC 19773 module 16? Why do we say 'may conform to ISO/IEC 19773' instead of 'shall conform'? What is the value of stating optional conformance?

A postal address enables the unambiguous determination of an actual or potential delivery point, usually combined with the specification of an addressee and/or a mailee. The data structure of Postal address may conform to ISO/IEC 19773 Information technology - Metadata registries (MDR) Modules - Module 16: Data Structure for UPU postal data.

### 5.1.13 Reference_Document

#### 5.1.13.1 Description of *Reference_Document*

A *Reference_Document* is a document that provides pertinent details for consultation about a subject.

The attributes of the *Reference_Document* class are summarized here and specified more formally in 5.1.13.2.

The following attributes are Mandatory:

— Every *Reference_Document* shall have exactly one *reference_document_identifier* of type *String,* which is an unambiguous identifier for the document.

— Every *Reference_Document* shall have exactly one *reference_document_type_description* of type *Text,* which describes the type of the document.

— Every *Reference_Document* shall have one or more *reference_document_provider*s of type *Organization, each of* which shall identify an *Organization* that maintains or carries an official copy of the *Reference_Document*.

The following attributes are Optional:

— A *Reference_Document* may have zero, one or more *reference_document_language_identifier*s, which specify the language or languages used in the *Reference_Document.*

— A *Reference_Document* may optionally have a *reference_document_notation* of type *Notation.*

— A *Reference_Document* may optionally have a *reference_document_title* of type *Text.*

— A *Reference_Document* may optionally have a *reference_document_uri* of type *String*.

#### 5.1.13.2 Attributes of *Reference_Document*

##### 5.1.13.2.1 reference_document_identifier

| | |
|---|---|
| Attribute name: | ***reference_document_identifier*** |
| Definition: | *Identifier* for the *Reference_Document* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *String* (5.1.16) |

##### 5.1.13.2.2 reference_document_type_description

| | |
|---|---|
| Attribute name: | ***reference_document_type_description*** |
| Definition: | description of the type of *Reference_Document* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Text* (5.1.17) |

##### 5.1.13.2.3 reference_document_language_identifier

| | |
|---|---|
| Attribute name: | ***reference_document_language_identifier*** |
| Definition: | *Identifier* of the natural language used in the *Reference_Document.* |
| Obligation: | Optional |
| Multiplicity: | *0..** |
| Data type: | *Language_Identification* (5.1.7) |
| Note: | Absence of a *reference_document_language_identifier* implies use of the language specified by *Registry_Specification registry_primary_language.* |

##### 5.1.13.2.4 reference_document_notation

| | |
|---|---|
| Attribute name: | ***reference_document_notation*** |
| Definition: | notation used within the reference document |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Notation* (5.1.9) |

##### 5.1.13.2.5 reference_document_title

| | |
|---|---|
| Attribute name: | ***reference_document_title*** |
| Definition: | title of the *Reference_Document.* |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Text* (5.1.17) |

##### 5.1.13.2.6 reference_provider

| | |
|---|---|
| Attribute name: | ***reference_provider*** |
| Definition: | *Organization* that maintains or carries an official copy of the *Reference_Document*. |
| Obligation: | Mandatory |

Multiplicity: *1..**

Data type: *Organization* (5.1.10)

#### 5.1.13.2.7 reference_document_uri

Attribute name: ***reference_document_uri***

Definition: uri for *Reference_Document*

Obligation: Optional

Multiplicity: *0..1*

Data type: *String* (5.1.16)

—— End of attributes of *Reference_Document* ——

### 5.1.14 Registration_Authority_Identifier

#### 5.1.14.1 Description of *Registration_Authority_Identifier*

The composite datatype *Registration_Authority_Identifier* is used to uniquely identify a *Registration_Authority*. The sources of values for each part of the identifier are specified in ISO/IEC 6523-1 and further explained for the metadata registry in ISO/IEC 11179-6.

A *Registration_Authority_Identifier* consists of the following parts, which are summarized here and specified more formally in 5.1.14.2.

The following parts are mandatory:

— Every *Registration_Authority_Identifier* shall have exactly one *international_code_designator* of type *String*.

— Every *Registration_Authority_Identifier* shall have exactly one *organization_Identifier* of type *String*.

The following parts are optional:

— A *Registration_Authority_Identifier* may optionally have an *organization_part_identifier (OPI)* of type *String*.

— A *Registration_Authority_Identifier* may conditionally have an *OPI_source* of type *String*. If the *organization_part_identifier* is present, then the *OPI_source* shall be present.

#### 5.1.14.2 Attributes of *Registration_Authority_Identifier*

#### 5.1.14.2.1 international_code_designator

Attribute name: ***international_code_designator***

Definition: *Identifier* of an organization identification scheme

Obligation: Mandatory

Multiplicity: *1*

Data type: *String* (5.1.16)

#### 5.1.14.2.2 organization_identifier

Attribute name: ***organization_identifier***

Definition: *Identifier* assigned to an *Organization* within an organization identification scheme, and unique within that scheme

Obligation: Mandatory

Multiplicity: *1*

Data type: *String* (5.1.16)

#### 5.1.14.2.3 organization_part_identifier (OPI)

Attribute name: ***organization_part_identifier***

Definition: *Identifier* allocated to a particular organization part

Obligation: Optional

Multiplicity: *0..1*

Data type: *String* (5.1.16)

#### 5.1.14.2.4 OPI_source

Attribute name: ***OPI_source***

Definition: the source for the *organization_part_identifier*

Obligation: Conditional

Multiplicity: *0..1*

Data type: *String* (5.1.16)

Condition: If *organization_part_identifier* is present, then *OPI_source* shall be present.

—— End of attributes of *Registration_Authority_Identifier* ——

### 5.1.15 Sign

A *sign* may be a character string, graphic image, sound clip or other symbol that can be used to denote or designate a *concept*. The *Sign* datatype may be represented by various expressions such as character string, sentence, code, icon, and so on.

### 5.1.16 String

[Character]string is a family of datatypes which represent strings of symbols from standard character-sets. The syntax and semantics of the String datatype are as defined in ISO/IEC 11404:2007 10.1.5 Character String.

### 5.1.17 Text

EDITOR'S NOTE #11. (Action required) CD19773 uses the term 'multitext' with the same definition as this. We should be consistent, or explain why we are inconsistent. More detail is required to explain the structure of the set of values within *Text*. Should we reference 19773 multitext?

*Text* is a set of textual values that all have the same meaning, but may have different representations and different datatypes that are dependent upon the context of use.

*Text* is data in the form of characters, symbols, words, phrases, paragraphs, sentences, tables, or other character arrangements, intended to convey a meaning, and whose interpretation is essentially based upon the reader's knowledge of some natural language or artificial language [ISO/IEC 2382-23:1994]

EXAMPLE A business letter printed on paper or displayed on a screen.

### 5.1.18 Value

EDITOR'S NOTE #12. (Action required) CD19773 uses the term 'multivalue' with the same definition as this. We should be consistent.

EDITOR'S NOTE #13. (Action required) Since the definition below references 'multidata' does that also need to be included as a defined datatype.

A *Value* is a set of values that all have the same meaning, but may have different representations and different datatypes that are dependent upon the context of use. Value represents a datatype that conforms to the semantics of the contextualized-value portion “multidata” datatype as described in 19773-03. A *Value* is a list of *Contextualized-Values*, each of which is a combination of “context-designation” that determines both the value space and the operation set of the actual value and an octet string and that represents the value itself.

See: [ISO/IEC 11404:2007 10.1.9 Private] for further explanation.

EDITOR'S NOTE #14. (Action required) 11404 and 19773 both have similar definitions for the contextualized value (Private in 11404). Do we really want to spec *sets* vs. individual entries? Do we want to specify some sort of default (e.g. a single character string?

# 6 Identification, Designation and Definition Package

## 6.1 Identification region

### 6.1.1 Overview

The Identification region of the model specifies how *metadata items* are identified in the *metadata registry*. A *metadata item* that is to be identified shall be assigned the type *Identified_Item*.

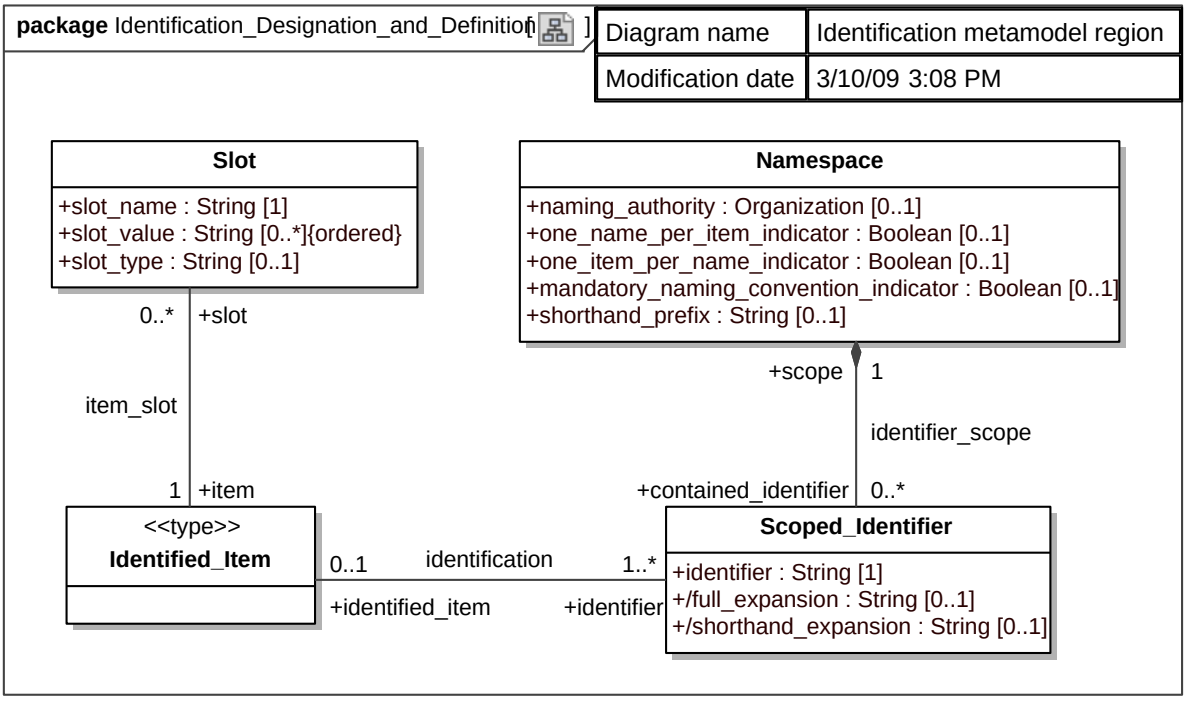



**Figure 6-1 — Identification metamodel region**

### 6.1.2 Classes in the Identification metamodel region

#### 6.1.2.1 Identified_Item

An *identified item* is a *metadata item* that is identified in a *metadata registry*. *Identified_Item* is a class that represents an *identified item.*

*Identified_Item* shall participate in the following association:

— *identification* (6.1.3.1) which references one or more *Scoped_Identifiers* (6.1.2.2), each of which provides an *identifier* (6.1.2.2.2.1) for the *Identified_Item* within a specific *Namespace* (6.1.2.3).

*Identified_Item* may participate in the following association:

— *item_slot* (6.1.3.3) which references zero or more *Slot*s (6.1.2.4) which extend the *Identified_Item*.

*Registered_Item* (7.1.2.1) is a subtype of *Identified_Item*. *Identified_Item* has no attributes.

#### 6.1.2.2 Scoped_Identifier

##### 6.1.2.2.1 Description of *Scoped_Identifier*

A *scoped identifier* is an *identifier* (6.1.2.2.2.1) with a particular scope, provided by a *Namespace* (6.1.2.3). *Scoped_Identifier* is a class that represents a *scoped identifier*.

*Scoped_Identifier* shall participate in the following association:

— *identifier_scope* (6.1.3.2) which references a *Namespace* which provides the scope for the *Scoped_Identifier*.

*Scoped_Identifier* may participate in the following association:

— *identification* (6.1.3.1) which references zero or one *Identified_Items* (6.1.2.1), which are unambiguously identified by the *Scoped_Identifier* within the *Namespace*.

The attributes of the *Scoped_Identifier* class are summarized here and specified more formally in 6.1.2.2.2.

— *Scoped_Identifier* shall have exactly one *identifier* of type *String* (5.1.16), which may be used as an unambiguous identifier for an *Identified_Item* within a particular *Namespace*.

EDITOR'S NOTE #15. '(Action required) How is full_expansion derived? Should Namespace be required to have exactly one identifier within its own scope that would be used as the prefix in the full expansion? If so, why is full_expansion optional? Should we add *full_prefix* as an attribute of *Namespace?*

— *Scoped_Identifier* may have zero or one derived attribute *full_expansion* (6.1.2.2.2.2), formed by prefixing the primary *identifier* of *Namespace* with the *identifier* of this *Scoped_Identifier.*

— *Scoped_Identifier* may have zero or one derived attribute *short_expansion* (6.1.2.2.2.3), formed by prefixing the *shorthand_prefix* (6.1.2.3.2.5) of *Namespace* with the *identifier* of this *Scoped_Identifier. short_expansion* will exist if and only if the corresponding *shorthand_prefix* exists.

##### 6.1.2.2.2 Attributes of *Scoped_Identifier*

###### 6.1.2.2.2.1 identifer

| | |
|---|---|
| Attribute name: | ***identifier*** |
| Definition: | *String* used to unambiguously denote an *Identified_Item* within the scope of a specified *Namespace.* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *String* (5.1.16) |

###### 6.1.2.2.2.2 full_expansion

| | |
|---|---|
| Attribute name: | ***full_expansion*** |
| Definition: | *String* representation of a *Scoped_Identifier*, in which the unique *identifier* of the associated *Namespace* is combined in some way with the *identifier* of the *Scoped_Identifier* to fully specify the scope. |
| Obligation: | Optional, derived. |
| Multiplicity: | *0..1* |
| Data type: | *String* (5.1.16) |
| Comment: | The manner of derivation is not specified, and will vary depending on the type of namespace. |

#### 6.1.2.2.2.3 shorthand_expansion

| | |
|---|---|
| Attribute name: | ***shorthand_expansion*** |
| Definition: | *String* representation of a *Scoped_Identifier* in which a *shorthand_prefix* from the associated *Namespace* has been prepended to the *identifier* to indicate the *scope*. |
| Obligation: | Conditional, derived. |
| Multiplicity: | *0..1* |
| Data type: | *String* (5.1.16) |
| Condition: | *short_expansion* will exist if and only if the corresponding *shorthand_prefix* (6.1.2.3.2.5) exists. |

—— End of attributes of *Scoped_Identifier* ——

### 6.1.2.3 Namespace

#### 6.1.2.3.1 Description of *Namespace*

A *Namespace* is a scoping construct used to group sets of *Designation*s (6.2.2.3) and/or *Scoped_Identifier*s (6.1.2.2) used in a metadata registry. Distinct *Namespaces* permit independent development of metadata collections and/or ontologies. They permit enforcement of uniqueness constraints on *identifier*s (6.1.2.2.2.1) or *designation_sign*s (6.2.2.3.2.1) within a specific *Namespace* without central coordination.

A *Namespace* may contain a set of *Designations,* a set of *Scoped_Identifier*s or a combination of the two.

NOTE These are NOT XML Namespaces. However, it may be possible to add additional subtypes of Namespaces to model XML Namespaces.

As a metadata item itself, a *Namespace* may be assigned a type of:

— *Identified_Item* (6.1.2.1)*,* enabling it to be identified;

— *Designatable_Item* (6.2.2.1), enabling it to be named and/or defined;

— *Classifiable_Item* (8.2.2.1), enabling it to be classified.

The attributes of the *Namespace* class are summarized here and specified more formally in 6.1.2.3.2.

— *Namespace* may have zero or one *naming_authority* (6.1.2.3.2.1) that specifies the *Organization* (5.1.10) that has authority for naming in the *Namespace*.

— *Namespace* may have zero or one *one_name_per_item_indicator* (6.1.2.3.2.2), signifying whether or not:
many *Designations* from the *Namespace* may be associated with (bound to) one *Designatable_Item,* and
many *Scoped_Identifiers* from the *Namespace* may be associated with (bound to) one *Identified_Item.*
If the *one_name_per_item_indicator* is null, then the rule is unspecified.

— *Namespace* may have zero or one *one_item_per_name_indicator* (6.1.2.3.2.3), signifying whether or not:
a given *Designation* may denote many *Designatable_Items, and*
a given *Scoped_Identifier* must identify only a single *Identified_Item.*
If the *one_item_per_name_indicator* is null, then the rule is unspecified.

— *Namespace* may have zero or one *mandatory_naming_convention_indicator* (6.1.2.3.2.4) that determines whether or not all *Designations* in a *Namespace* have to conform to exactly one *Naming_Convention*.

— *Namespace* may have zero or one *shorthand_prefix* (6.1.2.3.2.5), used as shorthand for the *Namespace* in text intended for human consumption.

#### 6.1.2.3.2 Attributes of *Namespace*

##### 6.1.2.3.2.1 naming_authority

| | |
|---|---|
| Attribute name: | ***naming_authority*** |
| Definition: | *Organization* that has the authority for naming in the *Namespace*. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Organization* (5.1.10) |

##### 6.1.2.3.2.2 one_name_per_item_indicator

| | |
|---|---|
| Attribute name: | ***one_name_per_item_indicator*** |
| Definition: | indicator that denotes whether more than one *Designation* and/or *Scoped_Identifier* within the *Namespace* may be associated with any single item (*Designatable_Item* and/or *Identified_Item*). If the indicator is *true*, then at most one *Designation* and/or *Scoped_Identifier* within the *Namespace* may be associated with any single item. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Boolean* (5.1.2) |
| Comment: | If the indicator is *true*, then the registry shall enforce the rule for the *Namespace*. |

##### 6.1.2.3.2.3 one_item_per_name_indicator

| | |
|---|---|
| Attribute name: | ***one_item_per_name_indicator*** |
| Definition: | indicator that denotes whether the *Namespace* may contain more than one *Designation* and/or *Scoped_Identifier* having the same sign and/or identifier. If the indicator is *true*, then at most one *Designation* and/or *Scoped_Identifier* having the same sign and/or identifier is permitted within the *Namespace*. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Boolean* (5.1.2) |
| Comment: | If the indicator is *true*, then the registry shall enforce the rule for the *Namespace*. |

##### 6.1.2.3.2.4 mandatory_naming_convention_indicator

| | |
|---|---|
| Attribute name: | ***mandatory_naming_convention_indicator*** |
| Definition: | indicator specifying whether all *Designations* in this *Namespace* shall conform to one of the acceptable *Naming_Conventions.* |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Boolean* (5.1.2) |
| Note: | If *mandatory_naming_convention_indicator* is true:<br>(a) there must be at least one acceptable convention *Naming_Convention* associated with this *Namespace* in the *naming_convention_utilization* association, and<br>(b) every binding *Designation* must have a *naming_convention_conformance* association with one of the same *Naming_Conventions* used in part (a) |

above.

If *mandatory_naming_convention_indicator* is false, it is possible for a *Namespace* to be associated with zero or more acceptable conventions and/or a binding *Designation* to conform to more than one convention.

#### 6.1.2.3.2.5 shorthand_prefix

Attribute name: ***shorthand_prefix***

Definition: prefix conventionally used as shorthand for a namespace, for greater readability, in text for human consumption.

NOTE In the case of URL prefixes as defined in XML, a final colon (:) should be included here as part of the shorthand prefix.

Obligation: Optional

Multiplicity: *0..1*

Data type: *String* (5.1.16)

—— End of attributes of *Namespace* ——

### 6.1.2.4 Slot

#### 6.1.2.4.1 Description of *Slot*

*Slot* instances provide a dynamic way to add arbitrary attributes to instances of *Identified_Item* (6.1.2.1). A *Slot* instance is associated with exactly one *Identified_Item* instance, through the association *item_slot* (6.1.3.3). An *Identified_Item* instance may be associated with zero, one or more *Slot* instances. All *Slot* instances associated with a particular *Identified_Item* instance must have a distinct *slot_name* (6.1.2.4.2.1) to allow each *Slot* instance to be unambiguously identified.

For example, if a company wants to add a “copyright” attribute to each *Identified_Item* instance that it submits, it can do so by adding a slot with name “copyright” and value containing the copyrights statement.

The attributes of the *Slot* class are summarized here and specified more formally in 6.1.2.4.2.

— A *Slot* shall have exactly one *slot_name* of type *String* (5.1.16), that unambiguously identifies the *Slot* among several associated with an *Identified_Item.*

— A *Slot* may have zero, one or more *slot_value*s (6.1.2.4.2.2) of type *String*, that provide the value(s) associated with the *slot_name*.

— A *Slot* may have zero or one *slot_type* (6.1.2.4.2.3) of type *String*, that specifies the datatype to be used to interpret the *slot_value*.

#### 6.1.2.4.2 Attributes of *Slot*

##### 6.1.2.4.2.1 slot_name

Attribute name: ***slot_name***

Definition: *name* of the *Slot*

Obligation: Mandatory

Multiplicity: *1*

Data type: *String* (5.1.16)

##### 6.1.2.4.2.2 slot_value

Attribute name: ***slot_value***

| | |
|---|---|
| Definition: | value assigned to the *Slot* |
| Obligation: | Optional |
| Multiplicity: | *0..** |
| Data type: | *String* (5.1.16) (ordered) |

#### 6.1.2.4.2.3 slot_type

| | |
|---|---|
| Attribute name: | ***slot_type*** |
| Definition: | *datatype* of the *slot_value* |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *String* (5.1.16) |

—— End of attributes of *Slot* ——

### 6.1.3 Associations in the Identification region

#### 6.1.3.1 identification

The *identification* association specifies the *Scoped_Identifier* that identifies an *Identified_Item*.

*identification* has two roles:

— identifier (verb form: identifies) which references a *Scoped_Identifier*;

— identified_item (verb form: identified_by).which references an *Identified_Item*

Every *Identified_Item* (6.1.2.1) must have one or more *identification* associations with a *Scoped_identifier* (6.1.2.2) that provides an *identifier* (6.1.2.2.2.1) for the *Identified_Item*.

#### 6.1.3.2 identifier_scope

*identifier_scope* associates zero, one or more *Scoped_Identifier*s (6.1.2.2) with exactly one *Namespace* (6.1.2.3).

*identifier_scope* has two roles:

— scope (verb form: provides_scope_for) which references a *Namespace*;

— contained_identifier (verb form: contained_in) which references a *Scoped_Identifier*.

#### 6.1.3.3 item_slot

*item_slot* associates an *Identified_Item* (6.1.2.1) with zero, one or more *Slots* (6.1.2.4) which extend the *Identified_item*.

*item_slot* has two roles:

— item (verb form: extended by) which references an *Identified_Item;*

— slot (verb form: extends) which references a *Slot*.

## 6.2 Designation and Definition region

### 6.2.1 Overview

The Designation and Definition region is used to manage the *Designations* (6.2.2.3) and *Definitions* (6.2.2.4) of *Designatable_Items* (6.2.2.1) and the *Contexts* (6.2.2.5) for the *Designations* and *Definitions*. A *Designatable_Item* may have many *designation_sign*s (6.2.2.3.2.1) that will vary depending on discipline, locality, technology, etc. This sub-clause describes the classes, associations, and association classes of this region.

Figure 6-2 represents the Designation and Definition region. This region of the metamodel is based on, and is consistent with, terminological models developed by ISO/TC 37.

ISO/IEC 11179-4 provides rules and guidelines for the formulation of data definitions.

ISO/IEC 11179-5 provides naming and identification principles for *Designatable_Item*s within a *Context*.

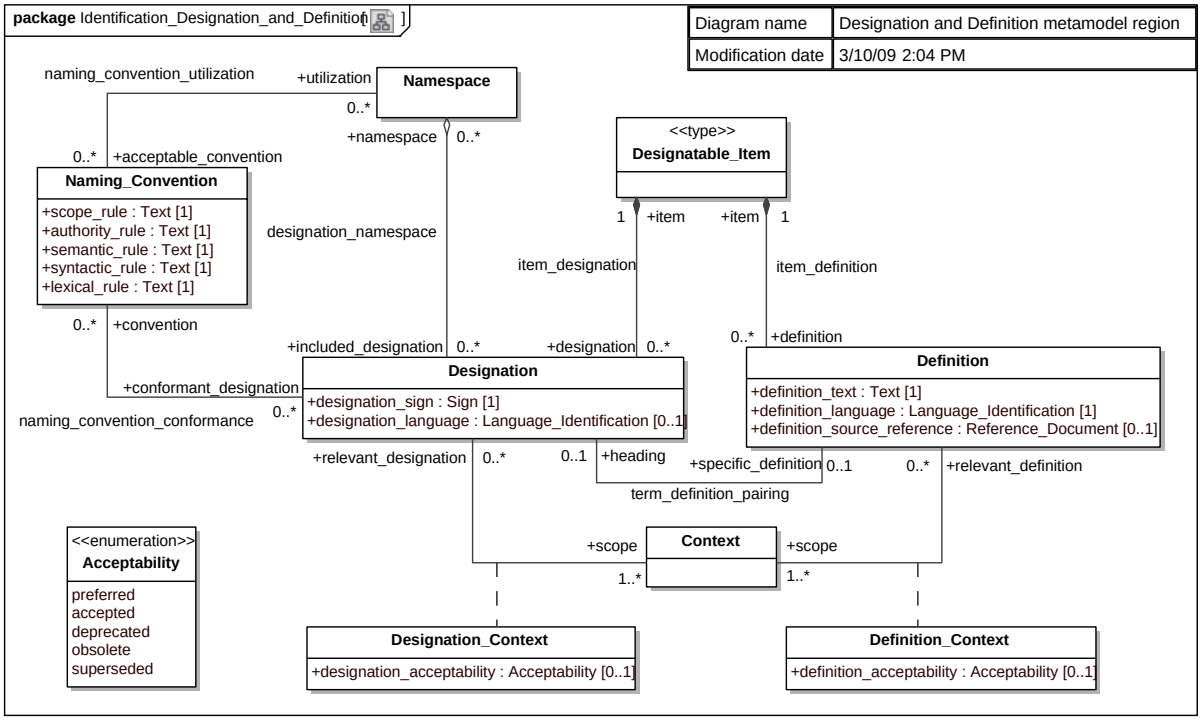



**Figure 6-2 — Designation and Definition metamodel region**

EDITOR'S NOTE #16. (Action required) The *term_definition_pairing* association uses the role 'heading' instead of 'term', because `term` is used by the *assertion_term* association. Should we rename term_definition_pairing to designation_definition_pairingÉ

EDITOR'S NOTE #17. '(Action required) It has been suggested that *term_definition_pairing* may be dependent on *Context*. For example, different terms may be used for the same definition in different contexts, or different definitions may be applied to the same term in different contexts. The Editor suggests making *Term_Definition_Pairing* a class associated with exactly one *Designation* and one *Definition*, and associating it with zero or more *Contexts*. Designations and Definitions may be associated zero or more *Term_Definition_Pairings.* We could specify a constraint that a Context shall be specified if a Designation or Definition is associated with more then one Term_Definition_Pairing, but it might be better to leave this to the discretion of a Registration_Authority.

### 6.2.2 Classes in the Designation and Definition region

#### 6.2.2.1 Acceptability

A scale of acceptability ratings comprised of: preferred, admitted, deprecated, obsolete and superseded. [ISO 10241.]

#### 6.2.2.2 Designatable_Item

*Designatable_Item* is the class of objects which can have designations and definitions. While it is not necessary for all *Designatable_Items* to have a designation and/or definition in a metadata registry, a metadata registry must be able to support the association of designations or definitions with *Designatable_Items* should they actually exist. *Designatable_Items* may have many different (or identical) *signs* in various *languages*, *Contexts* (6.2.2.5), *Namespaces* (6.2.2.6) and *Naming_Conventions* (6.2.2.7).

A *Designatable_Item* may have zero or more *item_designation* associations (6.2.4.3) with *Designation*s (6.2.2.3).

A *Designatable_Item* may have zero or more *item_definition* associations (6.2.4.2) with *Definition*s (6.2.2.4).

#### 6.2.2.3 Designation

##### 6.2.2.3.1 Description of *Designation*

The *Designation* class records the binding of a pair comprised of a *sign* (*designation_sign* 6.2.2.3.2.1) and its *language* (*designation_language* 6.2.2.3.2.2) to a *Designatable_Item* (6.2.2.1). Each *Designation* is situated with respect to a *Context* (6.2.2.5), a *Naming_Convention* (6.2.2.7), a *Namespace* (6.2.2.6), and may be paired with a *Definition* (6.2.2.4).

A *Designation* shall participate in

— exactly one *item_designation* association (6.2.4.3) with a *Designatable_Item*;

— one or more *designation_context* associations (6.2.3.2) with a *Context*;

A *Designation* may participate in the associations:

— *designation_namespace* (6.2.4.1) with zero or more *Namespaces* that provide scope for the *Designation*;

— *naming_convention_conformance* (6.2.4.4) with zero or more *Naming_Conventions* that provide naming rules for the *Designation*;

— *term_definition_pairing* (6.2.4.6) with zero or one *Definition*.

The attributes of the *Designation* class are summarized here and specified more formally in 6.2.2.3.2.

— A *Designation* shall have exactly one *designation_sign* (6.2.2.3.2.1) attribute, of type *Sign* (5.1.15), which is used to designate a *Designatable_Item*, e.g., a name of an object or concept. The *designation_sign* may be a word or phrase in a natural language (as specified by the *designation_language*), or it may be an icon or other symbol.

NOTE In Edition 2, the term *name* was used for what is now called *sign*. This change in Edition 3 has been made to bring its terminology into conformity with ISO 1087 Part 1 (from ISO TC 37).

— A *Designation* may have zero or one *designation_language* attribute, of type *Language_Identification*, which is used to record the language or dialect in which the *designation_sign* (usually a name) is used, when the *designation_sign* has an associated language. Usually the language will refer to a natural human language.

EDITOR'S NOTE #18. (Action required) Is there a better location for the table below? Is it useful?

The table below illustrates the differences between a *Designation* and a *Scoped_Identifier* (see 6.1.2.1).

| | **Designation** | **Scoped_Identifier** |
|---|---|---|
| Scope | Namespace and Context, independently | Namespace |
| Occurrence | Many allowable per *Designatable_Item* | Many allowable per *Identified_Item* |
| Language dependent | Yes | No |
| Type | *Sign* (5.1.15) | *String* (5.1.16) |

**Table 6-1: Comparison of Designation to Scoped_Identifier**

#### 6.2.2.3.2 Attributes of *Designation*

##### 6.2.2.3.2.1 designation_sign

| | |
|---|---|
| Attribute name: | ***designation_sign*** |
| Definition: | *Sign* of the *Designation* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Sign* (5.1.15) |

##### 6.2.2.3.2.2 designation_language

| | |
|---|---|
| Attribute name: | ***designation_language*** |
| Definition: | *Language* or dialect in which the *Sign* (usually a *name*) is expressed |
| Obligation: | Conditional |
| Multiplicity: | *0..1* |
| Data type: | *Language_Identification* (5.1.7) |
| Condition | *designation_language* is conditional because it may not be applicable. If it is applicable, it must be specified. *designation_language* shall not default to *Registry_Specification.registry_primary_language*, as *definition_language* does. |

—— End of attributes of *Designation* ——

### 6.2.2.4 Definition

#### 6.2.2.4.1 Description of *Definition*

The *Definition* class provides the *definition_text* for a *Designatable_Item* (6.2.2.2) as it applies in zero, one or more *Contexts* (6.2.2.5). Each *Designatable_Item* may be associated with zero, one or more *Definitions*, each *Definition* being specified in a particular language.

The *Definition* class records the binding of a pair of *definition_text* and its *definition_language* to a *Designatable_Item*. The *definition_text* is a statement (commonly in a natural language) which specifies the meaning of the *Designatable_Item*. It may additionally record a *definition_source_reference* for the *definition_text*.

Varying *definition_languages*, *definition_contexts*, and/or *term_definition_pairings* may be associated with the set of *Definitions* of a *Designatable_Item*. Each *Definition* is associated with only one *Designatable_Item*.

A *Definition* shall participate in:

— exactly one *item_definition* association (6.2.4.2) with a *Designatable_Item*;

— one or more *definition_context* associations (6.2.3.1) with a *Context*;

A *Definition* may participate in:

— zero or more *term_definition_pairing* associations with a *Designation* (6.2.2.3).

The attributes of the *Definition* class are summarized here and specified more formally in 6.2.2.4.2.

The *Definition* class has the following attributes:

— A *Definition* shall have exactly one *definition_text* attribute of datatype *Text* which contains the text which constitutes the definition.

— A *Definition* may have zero or one *definition_language* attribute of datatype *Language Identifier*. The *definition_language* attribute records the language in which the definition_text is written.

EDITOR'S NOTE #19. (Action required) The datatype of *definition_text* has been changed from *String* to *Text* by Issue 18. Since the *Text* datatype inherently supports multiple languages, do we still need the explicit *definition_language* attribute here? Or should we revert to *String*?

— A *Definition* may have zero or one *definition_source_reference* attribute of datatype *Reference_Document*. The *definition_source_reference* attribute may be used to record the origin of the definition.

#### 6.2.2.4.2 Attributes of *Definition*

##### 6.2.2.4.2.1 definition_text

| | |
|---|---|
| Attribute name: | ***definition_text*** |
| Definition: | text of the *Definition* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Text* (see 5.1.17) |

##### 6.2.2.4.2.2 definition_language

| | |
|---|---|
| Attribute name: | ***definition_language*** |
| Definition: | *Language* used to write the *definition_text* |
| Obligation: | Conditional |
| Multiplicity: | *0..1* |
| Data type: | *Language_Identification* (see 5.1.7) |
| Condition | If *Registry_Specification.registry_primary_language* (see 7.1.2.9) is specified, then *definition_language* may be omitted, implying that the language is that specified by the *Registry_Specification.registry_primary_language.* If *Registry_Specification.registry_primary_language* is not specified, then *definition_language* must be specified. |

##### 6.2.2.4.2.3 definition_source_reference

| | |
|---|---|
| Attribute name: | ***definition_source_reference*** |
| Definition: | reference to the source from which the *definition_text* is taken |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Reference_Document* (see 5.1.13) |
| Condition | If *Registry_Specification.registry_primary_language* (see 7.1.2.9) is specified, then *definition_language* may be omitted, implying that the language is that specified by the *Registry_Specification.registry_primary_language.* If *Registry_Specification.registry_primary_language* is not specified, then *definition_language* must be specified. |

—— End of attributes of *Definition* ——

### 6.2.2.5 Context

A *Context* is a universe of discourse in which certain *Designations* (6.2.2.3) or *Definitions* (6.2.2.4) are used to designate or define a set of *Designatable_Items* (6.2.2.2). Each *Designatable_Item* may be designated and/or defined within zero, one or more *Contexts*.

A *Context* defines the scope within which the subject data has meaning. A *Context* may be a business domain, an information subject area, an information system, a database, file, data model, standard document, or any other environment determined by the owner of the registry. Each *Context* may itself be made a *Designatable_Item* within the registry and be given a *designation* and/or a *definition*.

A *Context* may have zero or more *Definition_Context* (6.2.3.1) associations with a *relevant_definition* of type *Definition* where the *Context* provides the *scope* of the associated *Definition*.

A *Context* may have zero or more *Designation_Context* (6.2.3.2) associations with a *relevant_designation* of type *Designation* where the *Context* provides the *scope* of the associated *Designation*.

NOTE The requirement that all *Designations* and *Definitions* be associated with at least one *Context* applies even when the item being designated and defined is a *Context*, and this does not create a problem of infinite regress. For example, one straightforward way to satisfy this requirement is to include, within each *Context*, the *Designation*(s) and *Definition*(s) for itself; another is to place all *Designation*s and *Definition*s for *Context*s in a "registry context" (including the *Designation*(s) and *Definition*(s) for the registry context itself).

#### 6.2.2.6 Namespace

*Namespace* is described in 6.1.2.3. The following additional statements apply to this region.

If the *mandatory_naming_convention_indicator* is true:

(a) there must be exactly one acceptable *Naming_Convention* associated with this *Namespace* in the *naming_convention_utilization* association, and

(b) every *included_designation Designation* must have a *naming_convention_conformance* association with the same *Naming_Convention* used in part (a) above.

If *mandatory_naming_convention_indicator* is false, it is possible for a *Namespace* to be associated with zero or more acceptable conventions and/or a *conformant_designation Designation* to conform to more than one convention.

A *Designation_Space* may have zero or more *designation_space_membership* associations with an *included_designation* of type *Designation* where the *Designation_Space* provides the *namespace* of the associated *Designation*.

One use of *Designation_Spaces* is to permit the *Designatable_Item* represented by a *designation_sign* to be uniquely determined for a sign within a particular *Designation_Space*.

A *Designation_Space* may participate in the associations: *naming_convention_utilization* and *designation_space_membership*.

If the *one_name_per_item_indicator* (in *Namespace)* is true (a.k.a. unique names), then each *Designatable_Item* within the *Designations* of the *Namespace* has exactly one *Designation* within this *Namespace*.

Unique names implies a functional mapping from *Designatable_Items* to *designation_signs*. In common parlance, no possibility of aliases exists. Thus two distinct *designation_signs* (names) within a *Namespace* must refer to separate *Designatable_Items*.

If the *one_name_per_item_indicator* is false, then each *Designatable_Item* within the *Designations* of the *Namepace* may have more than one *Designation* within this *Namespace*.

If the *one_item_per_name_indicator* attribute is true (a.k.a. unambiguous names), then there exists at most one *Designatable_Item* associated with each *Designation* in the *Namespace*.

Unambiguous names implies a functional mapping from *designation_signs* to *Designatable_Items*.

#### 6.2.2.7 Naming_Convention

##### 6.2.2.7.1 Description of *Naming_Convention*

EDITOR'S NOTE #20. (Action required) Although *Designations* can be associated with both Naming_Conventions and Contexts, there is currently no way to specify that a particular Naming_Convention applies to a Designation in a particular Context. If one were to associate Naming_Convention to a Context, this could imply one of two things depending on the cardinality of the association.
(1) A Context can have many Naming_Conventions
Question: How would one know if a particular name came from a particular naming convention?
(2) A Context can have only one Naming_Convention
This then requires that ALL names in this Context have this Naming_Convention. This then implies that:
1. ALL names in ALL languages use this Naming_Convention
(i.e. French, English, Korean, etc.)
2. ALL names in a language use this Naming_Convention
(i.e. preferred term and non-preferred terms [synonyms])
Question: Doesn't this seem to be overly restrictive and unrealistic?

The *Naming_Convention* class provides the specification by which the *designation_sign* (name) of a *Designation* is developed. The *Naming_Convention* class records a set of rules for constructing *designation_signs* (names) to designate *Designatable_Items*. *Naming_Convention*s may range in complexity from very simple to very complex. The semantic, syntactic, and lexical_rules may each have their own complexity.

As a metadata item itself, a *Naming_Convention* may be assigned a type of:

— *Identified_Item,* enabling it to be identified;

— *Designatable_Item*, enabling it to be named and/or defined;

— *Classifiable_Item*, enabling it to be classified.

The *Naming_Convention* class may participate in the associations: *naming_convention_utilization* and *naming_convention_conformance*.

The attributes of the *Naming_Convention* class are summarized here and specified more formally in 6.2.2.7.2.

— A *Naming_Convention* shall have exactly one *scope_rule* of type *Text*, by which the scope of the naming convention shall be specified.

— A *Naming_Convention* shall have exactly one *authority_rule* of type *Text*, by which the authority of the naming convention shall be specified.

— A *Naming_Convention* shall have exactly one *semantic_rule* of type *Text*, by which the semantics of the *designation_signs* conforming to the naming convention shall be specified.

— A *Naming_Convention* shall have exactly one *syntactic_rule* of type *Text*, by which the syntax of the *designation_signs* conforming to the naming convention shall be specified.

— A *Naming_Convention* shall have exactly one *lexical_rule* of type *Text*, by which the appearance of the *designation_signs* conforming to the naming convention shall be specified.

NOTE Part 5 of this standard has a more elaborate discussion of naming conventions.

##### 6.2.2.7.2 Attributes of *Naming_Convention*

###### 6.2.2.7.2.1 scope_rule

Attribute name: ***scope_rule***

| | |
|---|---|
| Definition: | rule specifying the range within which the *naming_convention* is in effect. |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Text* (see 5.1.17) |
| Note: | In terms of the metadata registry, the scope of a naming convention may be as broad or narrow as the *Registration_Authority*, or other authority, determines is appropriate. The scope should document whether the naming convention is descriptive or prescriptive. |

#### 6.2.2.7.2.2 authority_rule

| | |
|---|---|
| Attribute name: | ***authority_rule*** |
| Definition: | rule identifying the authority that assigns *designation_signs* (names) and/or enforces *naming conventions* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Text* (see 5.1.17) |
| Note: | Examples of authorities include information technology standards committees or nomenclature standardization bodies (e.g., in biology). |

#### 6.2.2.7.2.3 semantic_rule

| | |
|---|---|
| Attribute name: | ***semantic_rule*** |
| Definition: | rule specifying the meanings of parts of a *designation_sign* (name) and possibly separators that delimit them in a *Naming_Convention* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Text* (see 5.1.17) |
| Note: | The rule should specify whether or not names convey meaning, and if so how. |

#### 6.2.2.7.2.4 syntactic_rule

| | |
|---|---|
| Attribute name: | ***syntactic_rule*** |
| Definition: | rule specifying the arrangement of parts of a *designation_sign* (name) and the separators that delimit them in a *Naming_Convention* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Text* (see 5.1.17) |
| Note: | The arrangement may be specified as relative or absolute, or some combination of the two. Relative arrangement specifies parts in terms of other parts, e.g., a rule within a convention might require that a qualifier term must always appear before the part being qualified appears. Absolute arrangement specifies a fixed occurrence of the part, e.g., a rule might require that the property term is always the last part of a name. The syntactic_principle might also specify the syntactic forms of the name (noun phrase or verb phrase) and the parts of speech used to construct a name. |

#### 6.2.2.7.2.5 lexical_rule

| | |
|---|---|
| Attribute name: | ***lexical_rule*** |
| Definition: | rule specifying the appearance of *designation_signs* (names): preferred and non-preferred terms, synonyms, abbreviations, part length, spelling, permissible character set, case sensitivity, etc. [Derived from ISO/IEC 11179- |

| | |
|---|---|
| | 5] |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Text* (see 5.1.17) |
| Note: | The result of applying lexical_rules should be that all names governed by a specific naming convention have a consistent appearance. An example lexical principle might be the specification of the use of camelCase capitalization of words in a phrase which are concatenated together. |

—— End of attributes of *Naming_Convention* ——

### 6.2.3 Association Classes in the Designation and Definition Region

#### 6.2.3.1 Definition_Context

##### 6.2.3.1.1 Description of *Definition_Context*

The *Definition_Context* association class records the *Context* (6.2.2.5) in which a *Definition* (6.2.2.4) occurs.

The association has two roles:

— relevant_definition (verb form: includes_relevant_definition) which references a *Definition;*

— scope (verb form: occurs_in_scope) which references a *Context*.

A relevant_definition may occur within of zero or more scopes. A scope may include zero or more relevant_definitions.

*Definition_Context* may have zero or one d*efinition_acceptability*, specifying the acceptability of a particular *Definition* within the specified *Context.*

##### 6.2.3.1.2 Attributes of *Definition_Context*

###### 6.2.3.1.2.1 definition_acceptability

EDITOR'S NOTE #21. (Action required) Should we add the class word ‘rating’ to the attribute name?

| | |
|---|---|
| Attribute name: | ***definition_acceptabilty*** |
| Definition: | rating of the acceptability of the *Definition* in the specified *Context*. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Acceptability* (see 6.2.2.1) |

—— End of attributes of *Definition_Context* ——

#### 6.2.3.2 Designation_Context

##### 6.2.3.2.1 Description of *Designation_Context*

The *Designation_Context* association class records the *Context* (6.2.2.5) in which a *Designation* (6.2.2.3) occurs.

The association has two roles:

— relevant_designation (verb form: includes_relevant_designation) which references a *Designation* class;

— scope (verb form: occurs_in_scope) which references a *Context* class.

A relevant_designation may have zero or more scopes. A scope may include zero or more relevant_designations.

*Designation_Context* may have zero or one d*esignation_acceptability*, specifying the acceptability of a particular *Designation* within the specified *Context.*

#### 6.2.3.2.2 Attributes of *Designation_Context*

##### 6.2.3.2.2.1 designation_acceptability

| EDITOR'S NOTE #22. (Action required) Should we add the class word 'rating' to the attribute name? |
|---|

| | |
|---|---|
| Attribute name: | ***designation_acceptabilty*** |
| Definition: | rating of the acceptability of the *Designation* in the specified *Context*. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Acceptability* (see 6.2.2.1) |

—— End of attributes of *Designation_Context* ——

### 6.2.4 Associations in the Designation and Definition Region

#### 6.2.4.1 designation_namespace

*designation_namespace* is an association between a *Designation* and a *Namespace* that indicates that the *Designation* is bound to that *Namespace*. The association is used to record the *Namespace*s in which a *Designation* is valid.

The association has two roles: namespace (verb form: occurs_in_namespace) and binding (verb form: binds_to). The namespace role refers to a *Namespace*. The binding role refers to a *Designation*. Each namespace may have zero or more bindings. Each binding may occur in zero or more namespaces.

#### 6.2.4.2 item_definition

*item_definition* is an association between a *Designatable_Item* and a *Definition,* that indicates that the *Designatable_Item* is defined by *Definition.* The association records all of the definitions for a specific *Designatable_Item*.

The association has two roles: item (verb form: used_for_item) and definition (verb form: has_definition). The item role references a *Designatable_Item*. The definition role references a *Definition*. Each definition shall have exactly one item. Each item may have zero or more definitions.

A *Definition* is used for exactly one *Designatable_item*. Definitions may not be not be reused across multiple *Designatable_Item*s.

#### 6.2.4.3 item_designation

*item_designation* is an association between a *Designatable_Item* and a *Designation,* that indicates that the *Designatable_Item* is designated by the *Designation.* The association records all of the *Designation*s (sign + language pairs) of a *Designatable_Item.*

The association has two roles: item (verb form: used_for_item) and designation (verb form: has_designation). The item role references a *Designatable_Item.* The designation role references a *Designation*. Each designation shall be used for exactly one item. An item may have zero or more designations.

*Designation*s may not be not be reused across multiple *Designatable_Items*.

#### 6.2.4.4 naming_convention_conformance

*naming_convention_conformance* is an association between a *Designation* and a *Naming_Convention* indicating that the *Designation* conforms to the *Naming_Convention*. The association records which *Naming_Convention*s (if any) a particular *Designation* conforms to.

The association has two roles: convention (verb form: conforms_to) and conformant_designation (verb form: has_conformant_designation). The convention role refers to a *Naming_Convention*. The conformant_designation role refers to a *Designation*. Each conformant_designation may have zero or more conventions. Each convention may have zero or more conformant_designations.

#### 6.2.4.5 naming_convention_utilization

*naming_convention_utilization* is an association between a *Namespace* and a *Naming_Convention* indicating that the *Namespace* uses the *Naming_Convention* for its *Designations.* The association records the *Naming_Convention*s (if any) used by a *Namespace*.

The association has two roles: utilization (verb form: utilized_by) and acceptable_convention (verb form: accepted_convention). The utilization role refers to a *Namespace*. The acceptable_convention role refers to a *Naming_Convention*. Each utilization may utilize zero or one accepted_conventions. Each accepted_convention has zero or more utilizations.

#### 6.2.4.6 term_definition_pairing

*term_definition_pairing* is an association between a *Designation* and a *Definition.* The association is used to bind *Designations* to their associated *Definitions*.

NOTE The requirement that a *Designation* be associated with a *Designatable_Item*, means that it is not possible to use the registry simply to record terms and their definitions without reference to some item. It is anticipated that such a requirement would be met by the use of a *Concept_System* (8.1.2.2), where term is associated with a *Concept* (8.1.2.1).

EDITOR'S NOTE #23. '(Action required) The following paragraph needs work. Can we use Designation instead of Heading?

The association has two roles: heading (verb form: used_for_heading) and specific_definition (verb form: defined_as). The heading role refers to a *Designation*. The specific_definition role refers to a *Definition*. Each specific_definition may have zero or one headings. Each heading may have zero or one specific_definitions.

# 7 Registration Package

## 7.1 Registration metamodel region

### 7.1.1 Overview

The Registration region supports the registration of items in a registry. ISO/IEC 11179-6 further describes the registration of *Administered_Items* (7.1.2.2).

Figure 7-1 shows the classes, relationships, attributes and composite attributes that support Registration.

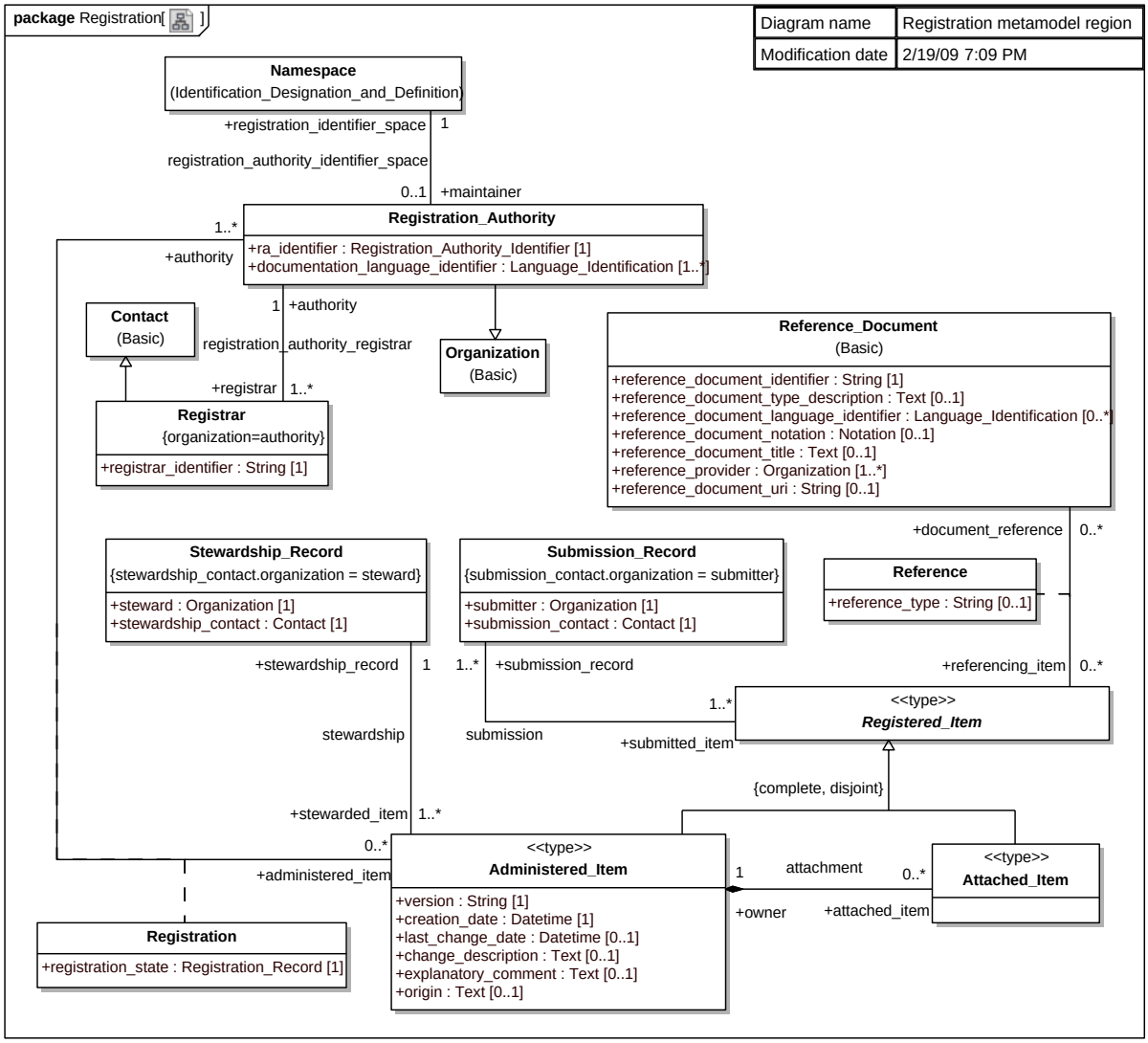



**Figure 7-1 — Registration metamodel region**

Figure 7-2 shows the *Registration_Record*, which records the state of a *Registration*.

EDITOR'S NOTE #24. (Action required) It has been suggested that *Registration_Record* be renamed to *Registration_State*. NB input is requested.

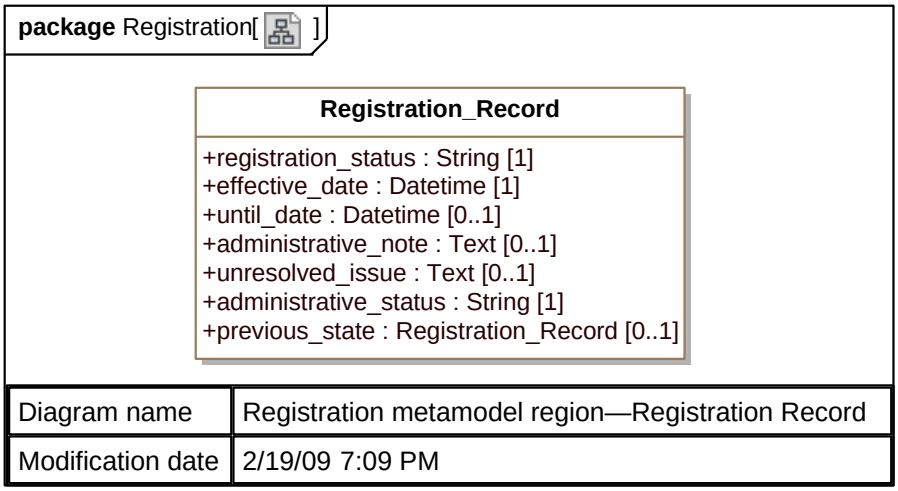



**Figure 7-2 — Registration_Record**

Figure 7-3 shows the *Registry_Specification*, which records information related to the registry as a whole.

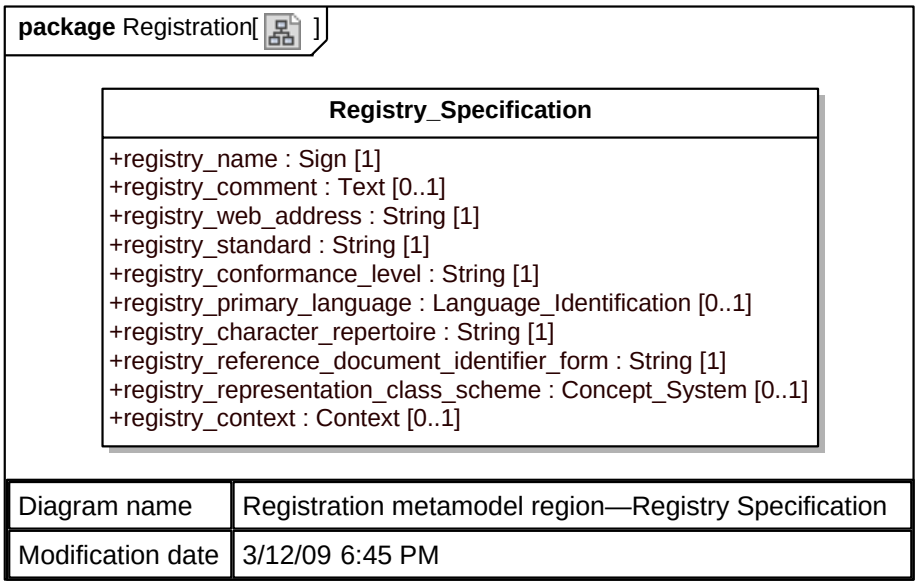



**Figure 7-3 — Registry specification**

### 7.1.2 Classes in the Registration region

#### 7.1.2.1 Registered_Item

*A Registered_Item* is an *Identified_Item* (6.1.2.1) that is registered and managed in a metadata registry.

A *Registered_Item* may also be a *Designatable_Item* (6.2.2.2), having one or more *Designations* and/or *Definitions*. A registration authority may specify that a *Registered_Item* is required to have at least one *Designation* and *Definition*.

A *Registered_Item* may also be a *Classifiable_Item* (8.2.2.1), associated with zero or more *Concepts* in one or more *Classification_Schemes*.

A *Registered_Item* must be either an *Administered_Item* (7.1.2.2) or an *Attached_Item* (7.1.2.3) but not both.

A *Registered_Item* must have a *submission* association (7.1.6.5) with one or more *Submission_Record*(s) (7.1.2.8) which identifies a *submitter* (7.1.2.8.2.1) of type *Organization* (7.1.3.2), and a *submission_contact* (7.1.2.8.2.2) of type *Contact* (7.1.3.1). The *submitter Organization* is the organization that has submitted the *Registered_Item* for addition, change or cancellation/withdrawal within a metadata registry. The *submission_contact* is the *Contact* at the *Organization* for issues related to the *submission.*

A *Registered_Item* may be described by zero or more *Reference_Documents* (7.1.3.3) as represented by the association class *Reference*(7.1.5.2).

### 7.1.2.2 Administered_Item

#### 7.1.2.2.1 Description of *Administered_Item*

An *Administered_Item* is a *Registered_Item* (7.1.2.1) for which administrative information is recorded. *Administered_Item* is a subtype of *Registered_Item*, that is administered by a *Registration_Authority* (7.1.2.5).

Every *Administered_Item* must have one or more *Registration* (7.1.5.1) associations with one or more *Registration Authorities*.

Every *Administered_Item* shall have exactly one *stewardship* association (7.1.6.4) with a *Stewardship_Record* (7.1.2.7), which identifies a *steward* (7.1.2.7.2.1) of type *Organization* (7.1.3.2) and a *stewardship_contact* (7.1.2.7.2.2) of type *Contact* (7.1.3.1), which is responsible for maintaining the item's administrative information.

An *Administered_Item* may have an *attachment* association (7.1.6.1) with zero or more *Attached_Items* (7.1.2.3). The set of *Attached_Items* that participate in this association are administered collectively under the one *Administered_Item* – they all share the same *stewardship*, *Registration*s and *version*. Note that *Attached_Items* may share the same *Administered_Item* and still have different *submitters* (7.1.2.8.2.1).

The attributes of the *Administered_Item* class are summarized here and specified more formally in 7.1.2.2.2.

EDITOR'S NOTE #25. (Action required) In response to Issue 176, it was agreed to move 'administrative_status' from Administration_Record to 'Registration', to allow different *Registration Authorities* to record different *administrative_statuses* for the same *Administered_Item*. However, it has also be suggested that (1) we should treat administrative (event) data orthogonally to registration (quality) data (e.g. as a separate association class), and (2) that we should have the same flexibility for all attributes in *Administration_Record*.

An *Administered_Item* contains:

— Exactly one *version* of type *String.* Whenever an *Administered_Item* is modified, the *version* identifier should be updated. Only one version of an *Administered_Item* can be registered at any one time within a particular *Namespace*, because *version* is not part of the identier. To register multiple versions concurrently, different *Namespaces* must be used.

— Exactly one *creation_date* of type *Date* that identifies the date and time that the *Administered_Item* was created.

— Zero or one *last_change_date* of type *Date* that specifies the date and time that the administered item was last changed.

— Zero or one *change descriptions* of type *Text* that describes what has changed in the *Administered_Item* since the prior version.

— Zero or one *explanatory_comment* of type *Text* that contains descriptive comments about the *Administered_Item*.

— Zero or one *origin* of type *Text* that describes the source (document, project, discipline or model)

#### 7.1.2.2.2 Attributes of *Administered_Item*

##### 7.1.2.2.2.1 version

Attribute name: ***version***

Definition: unique version *identifier* of the ***Administered_Item***

Obligation: Mandatory

Multiplicity: *1*

Data type: *String* (5.1.16)

##### 7.1.2.2.2.2 creation_date

Attribute name: ***creation_date***

Definition: date the ***Administered_Item*** was created

Obligation: Mandatory

Multiplicity: *1*

Data type: *Date* (5.1.4)

##### 7.1.2.2.2.3 last_change_date

Attribute name: ***last_change_date***

Definition: date the ***Administered_Item*** was last changed

Obligation: Optional

Multiplicity: *0..1*

Data type: *Date* (5.1.4)

##### 7.1.2.2.2.4 change_description

Attribute name: ***change_description***

Definition: description of what has changed since the prior *version* of the ***Administered_Item***

Obligation: Optional

Multiplicity: *0..1*

Data type: *Text* (5.1.17)

##### 7.1.2.2.2.5 explanatory_comment

Attribute name: ***explanatory_comment***

Definition: descriptive comments about the *Administered_Item*

Obligation: Optional

Multiplicity: *0..1*

Data type: *Text* (5.1.17)

#### 7.1.2.2.2.6 origin

| | |
|---|---|
| Attribute name: | ***origin*** |
| Definition: | the source (e.g. document, project, discipline or model) for the *Administered_Item* |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Text* (5.1.17) |

—— End of attributes of *Administered_Item* ——

### 7.1.2.3 Attached_Item

An *Attached_Item* is a *Registered_Item* (7.1.2.1) for which administrative information is recorded in another *Registered_Item* (an *Administered_Item* (7.1.2.2)). Every *Attached_Item* has an *attachment* (7.1.6.1) association with an *owning Administered_Item*, which supplies the administrative information.

*Attached_Item* provides a mechanism by which to administer a set of *Registered_Items* together, as a group, rather than maintaining separate administrative information for every individual item.

Example 1: the *Value_Meanings* within a *Conceptual_Domain* may be attached to, and thus administered with, the containing *Conceptual_Domain*.

Example 2: all the *Assertions* and *Concepts* within a *Concept_System* may be attached to, and thus administered with, the containing *Concept_System*.

### 7.1.2.4 Registrar

#### 7.1.2.4.1 Description of *Registrar*

*Registrar* is a subtype of *Contact* (7.1.3.1). *Registrar* is a *Contact* that is a representative of the *Registration_Authority* (7.1.2.5). A *Registration_Authority* is represented by one or more *Registrars*. A *Registrar* has a *registration_authority_registrar* association (7.1.6.3) with exactly one *authority Registration_Authority.*

*Registrars* are the persons who perform the administrative steps to register *Administered_Items* in a *Metadata Registry*.

*Registrar* has one mandatory attribute *registrar_identifier* of type *String* (5.1.16) that identifies a *Registrar*.

#### 7.1.2.4.2 Attributes of *Registrar*

##### 7.1.2.4.2.1 registrar_identifier

| | |
|---|---|
| Attribute name: | ***registrar_identifier*** |
| Definition: | identifier for the *Registrar* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *String* (5.1.16) |

—— End of attributes of *Registrar* ——

#### 7.1.2.5 Registration_Authority

##### 7.1.2.5.1 Description of Registration_Authority

A *Registration_Authority* is any *Organization* (7.1.3.2) responsible for maintaining a *register*. A *Registration_Authority* is a subtype of *Organization* and inherits all of its attributes and relationships.

A *Registration_Authority* may register many *Administered_Items* as shown by the *Regsitration* (7.1.5.1) association class.

A *Registration_Authority* shall have a *registration_authority_identifier_space* association (7.1.6.2) with a *registration_identifier_space Namespace* (7.1.4.1) that provides the scope for the *ra_identifier* (7.1.2.5.2.1) for the *Registration_Authority.*

The attributes of the *Registration_Authority* class are summarized here and specified more formally in 7.1.2.5.2.

— A *Registration_Authority* shall have an *ra_identifier* of type *Registration_Authority_Identifier*, which is the identifier for the *Registration_Authority.*

— A *Registration_Authority* shall have one or more *documentation_language_identifiers* of type *Language_Identification,* which specify the language(s) used for documentation by the *Registration_Authority.*

##### 7.1.2.5.2 Attributes of Registration_Authority

###### 7.1.2.5.2.1 ra_identifier

| | |
|---|---|
| Attribute name: | ***ra_identifier*** |
| Definition: | identifier of a *Registration_Authority* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Registration_Authority_Identifier* (5.1.14) |

###### 7.1.2.5.2.2 documentation_language_identifier

| | |
|---|---|
| Attribute name: | ***documentation_language_identifier*** |
| Definition: | identifier of the *Language* used for documentation by the *Registration_Authority* |
| Obligation: | Mandatory |
| Multiplicity: | *1..** |
| Data type: | *Language_Identification* (5.1.7) |
| Comment: | A registratoin authority may choose to default this attribute to *Registry_specification.registry_primary_language*. |

—— End of attributes of *Registration_Authority* ——

#### 7.1.2.6 Registration_Record

##### 7.1.2.6.1 Description of Registration_Record

A *Registration_Record* is a collection of information about the *Registration* (7.1.5.1) of an *Administered Item* (7.1.2.2).

The attributes of the *Registration_Record* class are summarized here and specified more formally in 7.1.2.6.2.

A *Registration_Record* shall have:

— a *registration_status* of type *String* which designates the status of an *Administered_Item* in the registration live-cycle;

— an *effective_date* is a *Date* that identifies the date and time that an *Administered_Item* became or will become available to registry users.

A *Registration_Record* may have:

— an *administrative_status* of type *String* which designates the status of an *Administered_Item* in the administrative process of a *Registration_Authority*;

— an *until_date* is a *Date* that identifies the date and time that an *Administered_Item* is or will no longer be effective in the registry;

— an *administrative_note* is a *Text* that contains general comments and instructions about the *Administered_Item;*

— an *unresolved_issue* is a *Text* that documents any problem that remains unresolved regarding proper documentation of the *Administered_Item*;

— a *previous_state* of type *Registration_Record* which records the immediately prior state of state of the registration.

#### 7.1.2.6.2 Attributes of Registration_Record

EDITOR'S NOTE #26. '(Action required) If the effective date applies to the Administered_Item, and not to the Registration, why is it in the Registration_Record instead of in Administered_Item?

##### 7.1.2.6.2.1 registration_status

| | |
|---|---|
| Attribute name: | ***registration_status*** |
| Definition: | designation of the status in the registration life-cycle of an *Administered_Item* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *String* (5.1.16) |
| Note: | Designation values are described in ISO/IEC 11179-6. |

##### 7.1.2.6.2.2 effective_date

| | |
|---|---|
| Attribute name: | ***effective_date*** |
| Definition: | date an *Administered_Item* became/becomes available to registry users |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Date* (5.1.4) |

##### 7.1.2.6.2.3 until_date

| | |
|---|---|
| Attribute name: | ***until_date*** |
| Definition: | date the *Registration* of an *Administered_Item* by a *Registration_Authority* in a registry is no longer effective |
| Obligation: | Optional |
| Multiplicity: | *0..1* |

Data type: *Date* (5.1.4)

#### 7.1.2.6.2.4 administrative_note

Attribute name: ***administrative_note***

Definition: general note(s) about the *Registration*

Obligation: Optional

Multiplicity: *0..1*

Data type: *Text* (5.1.17)

#### 7.1.2.6.2.5 unresolved_issue

Attribute name: ***unresolved_issue***

Definition: any problem(s) that remains unresolved regarding proper documentation of the *Administered_Item*

Obligation: Optional

Multiplicity: *0..1*

Data type: *Text* (5.1.17)

#### 7.1.2.6.2.6 administrative_status

Attribute name: ***administrative_status***

Definition: designation of the status in the administrative process of a *Registration_Authority*

Obligation: Optional

Multiplicity: *0..1*

Data type: *String* (5.1.16)

Note: The values and associated meanings of the *administrative_status* are determined by each *Registration_Authority*. C.f. *registration_status*.

#### 7.1.2.6.2.7 previous_state

Attribute name: ***previous_state***

Definition: immediately prior collection of administrative data about registration.

Obligation: Optional

Multiplicity: *0..1*

Data type: *Registration_Record* (7.1.2.6)

—— End of attributes of *Registration_Record* ——

### 7.1.2.7 Stewardship_Record

#### 7.1.2.7.1 Description of Stewardship_Record

The *Stewardship_Record* identifies both a *steward* (7.1.2.7.2.1) of type *Organization* (7.1.3.2), and a *stewardship_contact* (7.1.2.7.2.2) of type *Contact* (7.1.3.1) at the *Organization*, for one or more *Administered_Items* as represented by the s*tewardship* association (7.1.6.4) with *Stewardship_Record*.

The attributes of the *Stewardship_Record* class are summarized here and specified more formally in 7.1.2.7.2.

A *Stewardship_Record* shall have:

— a *steward* of type *Organization;*

— a *stewardship_contact* of type *Contact.*

#### 7.1.2.7.2 Attributes of Stewardship_Record

##### 7.1.2.7.2.1 steward

| | |
|---|---|
| Attribute name: | ***steward*** |
| Definition: | *Organization* that maintains *stewardship* of an *Administered_Item.* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Organization* (5.1.10) |

##### 7.1.2.7.2.2 stewardship_contact

| | |
|---|---|
| Attribute name: | ***stewardship_contact*** |
| Definition: | Contact information associated with a *Stewardship* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Contact* (5.1.3) |

—— End of attributes of *Stewardship_Record* ——

### 7.1.2.8 Submission_Record

#### 7.1.2.8.1 Description of Submission_Record

Each *Registered_Item* (7.1.2.1) must have a *submission* association (7.1.6.5) with one or more *Submission_Record*(s) which identifies a *submitter* (7.1.2.8.2.1) of type *Organization* (7.1.3.2), and a *submission_contact* (7.1.2.8.2.2) of type *Contact* (7.1.3.1). The *submitter Organization* is the organization that has submitted the *Registered_Item* for addition, change or cancellation/withdrawal within a metadata registry. The *submission_contact* is the *Contact* at the *Organization* for issues related to the *submission.*

NOTE A *Submission_Record* is required for *Attached_Items* as well as to *Administered_Items* because they can be submitted separately. However, one submission record can be used for multiple items submitted together.

The attributes of the *Submission_Record* class are summarized here and specified more formally in 7.1.2.8.2.

A *Submission_Record* shall have:

— a *submitter* of type *Organization;*

— a *submission_contact* of type *Contact.*

#### 7.1.2.8.2 Attributes of Submission_Record

##### 7.1.2.8.2.1 submitter

| | |
|---|---|
| Attribute name: | ***submitter*** |
| Definition: | *Organization* which has submitted a *Registered_Item* for inclusion in a register. |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Organization* (5.1.10) |

#### 7.1.2.8.2.2 submission_contact

| | |
|---|---|
| Attribute name: | ***submission_contact*** |
| Definition: | Contact information associated with a *Submission.* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Contact* (5.1.3) |

—— End of attributes of *Submission_Record* ——

### 7.1.2.9 Registry_Specification

#### 7.1.2.9.1 Description of Registry_Specification

*Registry_Specification* describes the environment in which the *Registry* operates.

The attributes of the *Registry_Specification* class are summarized here and specified more formally in 0.

A *Registry_Specification* shall have:

— a *registry_name* of type *Sign*

— a *registry_web_address* of type *String*

— a *registry_standard* of type *String*

— a *registry_conformance_level* of type *String*

— a *registry_character_repertoire* of type *String*

— a *registry_reference_document_identifier_form* of type *String.*

A *Registry_Specification* may have:

— a *registry_primary_language* of type *Language_Identification*

— a *registry_representation_class_scheme* of type *Concept_System*

— a *registry_context* of type *Context*

— a *registry_comment* of type *Text*

#### 7.1.2.9.2 Attributes of Registry_Specification

#### 7.1.2.9.2.1 registry_name

| | |
|---|---|
| Attribute name: | ***registry_name*** |
| Definition: | name by which the *Registry* is commonly known. |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Sign* (5.1.15) |
| Example: | US EPA Environmental Data Registry |

#### 7.1.2.9.2.2 registry_web_address

Attribute name: ***registry_web_address***

Definition: The World Wide Web uniform resource locator (url) for the registry.

Obligation: Mandatory

Multiplicity: *1*

Data type: *String* (5.1.16)

Example: www.epa.gov/edr

#### 7.1.2.9.2.3 registry_standard

Attribute name: ***registry_standard***

Definition: The standard to which this registry complies.

Obligation: Mandatory

Multiplicity: *1*

Data type: *String* (5.1.16)

Example: ISO/IEC 11179-3:2003 with Cor 1:2004

#### 7.1.2.9.2.4 registry_conformance_level

Attribute name: ***registry_conformance_level***

Definition: The conformance level of the registry as described in the standard.

Obligation: Mandatory

Multiplicity: *1*

Data type: *String* (5.1.16)

Example: Conformance level 2

#### 7.1.2.9.2.5 registry_character_repertoire

Attribute name: ***registry_character_repertoire***

Definition: The character reperoire that is used by the registry for internal operation.

Obligation: Mandatory

Multiplicity: *1*

Data type: *String* (5.1.16)

Example: ISO/IEC 646:1991 orISO/IEC 10646-1:2000

#### 7.1.2.9.2.6 registry_reference_document_identifier_form

Attribute name: ***registry_reference_document_identifier_form***

Definition: Specification of the form of identifier used for identifying Reference_Documents in the registry. Some registries may use URIs. Other registries may utilize an external document management system which provides unstructured, opaque identifiers. Yet other registries may define a structured identifier form which embeds an identifier type within each identifier in the registry. This attribute specifies the form of identifiers used for Reference_Documents in this particular registry.

Obligation: Mandatory

Multiplicity: *1*

Data type: *String* (5.1.16)

#### 7.1.2.9.2.7 registry_primary_language

Attribute name: ***registry_primary_language***

| | |
|---|---|
| Definition: | The primary and/or default language that is used for the registry. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Language_Identification* (5.1.7) |
| Example: | *eng-840 or en-US* |

#### 7.1.2.9.2.8 registry_representation_class_scheme

| | |
|---|---|
| Attribute name: | ***registry_representation_class_scheme*** |
| Definition: | *Concept_System* used by the registry to capture representation classes. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Concept_System* (8.1.2.2) |
| Note: | Edition 2 had a separate structure to record representation classes. In Edition 3, these are considered to be just another classification scheme, which in turn is considered a *Concept_System*. This attribute allows the registry to specify which *Concept_System* is used for this purpose. |

#### 7.1.2.9.2.9 registry_context

| | |
|---|---|
| Attribute name: | ***registry_context*** |
| Definition: | *Context* which represents the *Registry* itself. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Context* (6.2.2.5) |
| Comment: | It may sometimes be useful to reference a default *Context*, rather than create a new one. The *registry_context* is one possible choice for such a default. |

#### 7.1.2.9.2.10 registry_comment

| | |
|---|---|
| Attribute name: | ***registry_comment*** |
| Definition: | any comment that should be noted about the registry. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Text* (5.1.17) |

—— End of attributes of *Registry_Specification* ——

### 7.1.3 Classes referenced from the Basic package

#### 7.1.3.1 Contact

*Contact* is the class of object to whom information item(s), material object(s) and/or person(s) can be sent to or from. *Contact* is described in 5.1.3. *Registrar* (see 7.1.2.4) is a subtype of *Contact*.

#### 7.1.3.2 Organization

*Organization* is a unique framework of authority within which *individuals* act, or are designated to act, towards some purpose.

An *Organization* can play one or more roles with respect to a Metadata Registry. All *Registration Authorities* are *Organizations*, but not all *Organizations* are necessarily *Registration Authorities*. An *Organization* may be a *submitter* within a *Submission_Record,* with zero or more *submitted item Registered_Items*, where the

*Organization* acts as the *submitter.* An *Organization* may also have a *stewardship* association with zero or more *stewarded item Administered_Items* where the *Organization* acts as the *steward* for the item.

Organization is further described in 5.1.10.

#### 7.1.3.3 Reference_Document

*Reference_Document* is a document that provides pertinent details for consultation about a subject. *Reference_Document* is specified in 5.1.13.

A *Registered_Item* (7.1.2.1) may reference one or more *Reference Documents* as shown by the association class *Reference* (7.1.5.2) in Figure 7-1.

### 7.1.4 Classes referenced from the Identification, Designation and Definition package

#### 7.1.4.1 Namespace

A *Registration_Authority* (7.1.2.5) shall have a *registration_authority_identifier_space* association (7.1.6.2) with a *Namespace* that provides the *registration_identifier_space* for the *Registration_Authority. Namespace* is described in 6.1.2.3.

### 7.1.5 Association Classes in the Registration region

#### 7.1.5.1 Registration

##### 7.1.5.1.1 Description of Registration

EDITOR'S NOTE #27. (Action Required) Now that we have introduced Registered_Item as a supertype of Administered_Item, it would seem that the 'registration' association should be with 'Registered_Item', and a separate 'administration' association with 'Administered_Item', or, if the association name is considered correct, then the classes need to be renamed to better reflect their usage. We need a statement of requirements before we can correctly model this.

*Registration* is an association between a *Registration_Authority* (7.1.2.5) and an *Administered_Item* (7.1.2.2) where the *Registration_Authority* manages the *Administered_Item* in a metadata register.

*Registration* is also a class, and has an attribute *registration_state* of type *Registration_Record*, as specified below.

##### 7.1.5.1.2 Attributes of Registration

###### 7.1.5.1.2.1 registration_state

| | |
|---|---|
| Attribute name: | ***registration_state*** |
| Definition: | current collection of administrative data about *Registration* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Registration_Record* (7.1.2.6) |

—— End of attributes of *Registration* ——

#### 7.1.5.2 Reference

##### 7.1.5.2.1 Description of Reference

A *Reference* is the association between a *Reference_Document* (7.1.3.3) and a *Registered_Item* (7.1.2.1).

A *Reference* is also a class, and may have zero or one *reference_types* of type *String*.

#### 7.1.5.2.2 Attributes of Reference

##### 7.1.5.2.2.1 reference_type

| | |
|---|---|
| Attribute name: | ***reference_type*** |
| Definition: | specification of the type of *Reference* |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *String* (5.1.16) |

—— End of attributes of *Reference* ——

EDITOR'S NOTE #28. (Action required) What are some examples of reference_type? Is it a long reference vs a short reference; is it a legal reference vs a side note reference; is it an important reference vs an unimportant reference; etc? Why do we need it? If we do, fo interoperability, we should provide either candidate meanings and associated values, or ways to reference standardized types. Should the datatype be Text instead of String, to allow the type to be expressed in multiple languages?

### 7.1.6 Associations in the Registration region

#### 7.1.6.1 attachment

*attachment* is an association between an *Administered_Item* (7.1.2.2) and an *Attached_Item* (7.1.2.3) that indicates that the *Attached_Item* shares all of the administration characteristics of the *Administered_Item*. *attachment* enables collections of *Registered_Items* (7.1.2.1) to be administered collectively as a block.

*Attachment* has two roles: *owner* (verb form: *has owner*) and *attached item* (verb form: *attached to*). The *owner* role references an *Administered_Item* and the *attached item* role references an *Attached_Item*. Every *Attached_Item* shall have an *attachment* association with exactly one *owner Administered_Item*. An *Administered_Item* may have an *attachment* association with zero or more *Attached_Items*.

#### 7.1.6.2 registration_authority_identifier_space

*registration_authority_identifier_space* is an association between a *Registration_Authority* (7.1.2.5) and a *Namespace* (7.1.4.1). Every *Registration_Authority* shall have one *registration_authority_identifier_space* with a *registration_identifier_space Namespace* that provides the scope for the *ra_identifier* (7.1.2.5.2.1) for the *Registration_Authority.*

#### 7.1.6.3 registration_authority_registrar

*registration_authority_registrar* is an association between a *Registration_Authority* (7.1.2.5) and a *Registrar* (7.1.2.4) that indicates that the *Registrar* is a representative of the *Registration_Authority*.

Every *Registration_Authority* shall have one or more *registration_authority_registrar* associations with a *Registrar* where the *Registration_Authority* provides the authority of the associated *Registrar.*

#### 7.1.6.4 stewardship

*stewardship* is the association of an *Administered_Item* (7.1.2.2) to a *Stewardship_Record* (7.1.2.7), which records the *steward* (7.1.2.7.2.1) of type *Organization* (7.1.3.2) and *stewardship_contact* (7.1.2.7.2.2) of type *Contact* (7.1.3.1) involved in the stewardship of the *Administered_Item*.

#### 7.1.6.5 submission

*submission* is the association of a *Registered_Item* (7.1.2.1) with a *Submission_Record* (7.1.2.8), which records the *submitter* (7.1.2.8.2.1) of type *Organization* (7.1.3.2) involved in the *submission* of the *Registered_Item.*

# 8 Concepts Package

## 8.1 Concept System region

### 8.1.1 Overview

The Concept System metamodel region is illustrated in Figure 8-1. The purpose of the Concept System Metamodel Region is to describe *Concept*s (8.1.2.1) (abstract units of knowledge) and the various *Relation*s (8.1.2.4) which may hold among Concepts. Ontologies are supported as *Concept_Systems* with formal semantics through the use of *Assertions* (8.1.2.3).

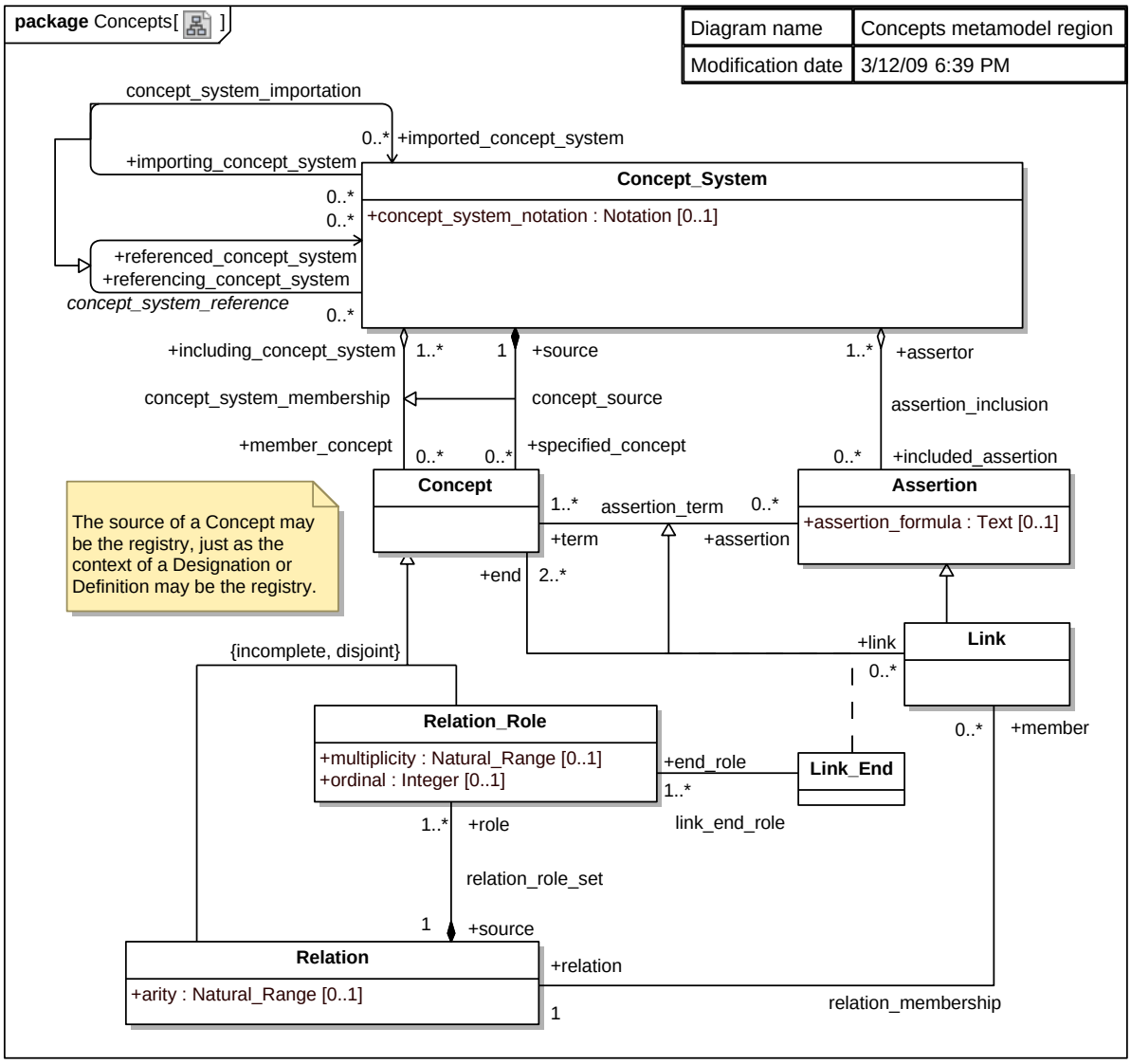



**Figure 8-1 — Concept system metamodel region**

### 8.1.2 Classes in the Concept_System region

#### 8.1.2.1 Concept

A *concept* is unit of knowledge created by a unique combination of characteristics. It is independent of representation. *Concept* is a class which represents a *concept.*

A *Concept* shall participate in the following associations:

— *concept_system_membership* (8.1.3.1) by which zero or more *Concepts* may be included in one or more *Concept_Systems* (8.1.2.2). Each *Concept* shall be a member of at least one *Concept_System*.

Note: The *Registry* may be specified as a *Concept_System* if no more appropriate *Concept_System* is defined.

— *concept_source* (8.1.3.2) by which exactly one *Concept_System* shall be specified as the source of each *Concept.*

A *Concept* may participate in the following associations:

— *Link_End* (8.1.4.1) with zero or more *Links* (8.1.2.6) where each *Concept* represents an *end* of the *Link*.

Note: a *Link* can have two or more Link*_Ends*, depending on the *arity* of the *relation*.

Several of the classes in the Data Description package (10) are subtypes of *Concept – see* Figure 10-7 — Types of Concepts in the Data Description package on p.108.

#### 8.1.2.2 Concept_System

##### 8.1.2.2.1 Description of Concept_System

A *concept system* is a set of *concepts* structured according to the *relations* among them. It is used to describe a domain of discourse.. *Concept_System* is a class which represents a *concept system.*

A minimal *concept system* could simply be a collection of *concepts*. A more elaborate *concept system* could be a collection of *concepts* which may be organized into a taxonomy or partonomy specified by means of various *relations* (e.g., semantic relations) and *links* amongst the *concepts*.

A much more elaborate subtype of *concept system* might be an (axiomatized) *ontology* specified by means of predicates and axioms among the concepts. Examples of *concept systems* are included in Annex F. The use of *concept systems* as *classification schemes* is described in clause 8.2.

A *Concept_System* may participate in the following associations:

— *concept_system_reference* (8.1.3.3) by which zero or more referenced *Concept_Systems* may be referenced by zero or more referencing *Concept_System*s.

— *concept_system_importation* (8.1.3.4) by which zero or more imported *Concept_Systems* may be imported into zero or more importing *Concept_System*s. *concept_system_importation* is a specialization of *concept_system_reference.*

— *concept_system_membership* (8.1.3.1) by which zero or more *Concepts* (8.1.2.1) may be included in one or more *Concept_Systems.* Each *Concept* shall be a member of at least one *Concept_System*. Since *Relations* (8.1.2.4) and *Relation_Roles* (8.1.2.5) are sub-types of *Concepts*, these too are included in one or more *Concept_Systems*.

— *concept_source* (8.1.3.2) by which exactly one *Concept_System* shall be specified as the source of each *Concept.*

— *assertion_inclusion* (8.1.3.5) by which zero or more *Assertions* (8.1.2.3) may be included in one or more *Concept_Systems.* Since a *Link* (8.1.2.6) is a subtype of *Assertion*, *Links* are also included in one or more *Concept_Systems.*

A *Concept_System* has exactly one *concept_system_notation* attribute of type *Notation*. The *concept_system_notation* attribute is used to record the *Notation* used to describe the *Concept_System*.

Examples of such notations include XCL Common Logic (ISO/IEC 24707) or OWL-DL XML notation (Ontology Web Language from W3C).

#### 8.1.2.2.2 Attributes of Concept_System

##### 8.1.2.2.2.1 concept_system_notation

| | |
|---|---|
| Attribute name: | ***concept_system_notation*** |
| Definition: | formal syntax and semantics used in the *concept system*. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Notation* (5.1.9) |

—— End of attributes of *Concept_System* ——

### 8.1.2.3 Assertion

#### 8.1.2.3.1 Description of Assertion

An *assertion* is a sentence or proposition in logic which is asserted (or assumed) to be true. *Assertion* is a class that represents an *assertion.*

*Assertion* shall participate in the following associations:

— *assertion_inclusion* (8.1.3.5) with one or more *Concept_Systems* (8.1.2.2) in an *assertor* role

— *assertion_term* (8.1.3.6) with one or more *Concepts* (8.1.2.1) in a *term* role

*Assertion* has one attribute, *assertion_formula* which expresses the *assertion*.

#### 8.1.2.3.2 Attributes of Assertion

##### 8.1.2.3.2.1 assertion_formula

| | |
|---|---|
| Attribute name: | ***assertion_formula*** |
| Definition: | text which expresses an *assertion* |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Text* (5.1.17) |

—— End of attributes of *Assertion* ——

#### 8.1.2.4 Relation

##### 8.1.2.4.1 Description of Relation

EDITOR'S NOTE #29. (Action required) Note 1 states that the *relation* is defined by means of a *predicate*, but no predicate is specified in the model. Without a predicate, how is the relevant subset specified? It seems inadequate to infer the relation from its member links. Since *Relation* is a subtype of *Concept*, it is possible we could use *Assertions* on the *Concept* to specify the *Relation*. If this is intended it should be explicitly stated, and note 1 needs to be reworded.

A *relation* is a subset of the powerset of RxUD, for some role set R, where UD is the universe of discourse. *Relation* is a class that represents a *relation.*

Note 1: An *n*-ary *relation* on sets $A_1$, ..., $A_n$ is a set of ordered *n*-tuples $<a_1, ..., a_n>$ where $a_i$ is an element of $A_i$ for all *i*, *i between 1 and n* . Thus an *n*-ary relation on sets $A_1$, ..., $A_n$ is a subset of Cartesian product $A_1$ x ... x $A_n$ . Membership of an n-tuple in the relation is specified by means of a predicate which must be true for the n-tuple to be a member of the corresponding relation. In this metamodel, *relations* are defined over sets of *concepts*.

Note 2: In this metamodel we actually use unordered n-tuples with named *Relation_Roles* rather than positional elements of the n-tuple. The ordering can optionally be specified using the *ordinal* attribute on *Relation_Role.*

*Relation* may participate in the following associations:

— *relation_roleset* (**Error! Reference source not found.**) with one or more *Relation_Roles* (8.1.2.5), each of which specifies the role of an element in the *relation*. The number of *Relation_Roles* is specified by the *arity* of the *relation*.

— *relation_membership* (8.1.3.7) with zero or more *Links* (8.1.2.6) as members of the *Relation.*

*Relation* is a supertype of the *Binary_Relation* class which is described in 9.1.2.2. *Relation* has one attribute *arity*, which specifies the number of elements in the *relation*.

##### 8.1.2.4.2 Attributes of Relation

###### 8.1.2.4.2.1 arity

| | |
|---|---|
| Attribute name: | ***arity*** |
| Definition: | number of elements in the relation |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Natural_Range* (5.1.8) |
| Example: | A *binary relation* has an arity of 2. |

—— End of attributes of *Relation* ——

#### 8.1.2.5 Relation_Role

##### 8.1.2.5.1 Description of Relation_Role

A *relation role* is an argument (element) of a *relation*. *Relation_Role* is a class which represents a *relation role.*

Note: In relational database terms, the *relation role* represents a column in a relational table (for an asymmetric relation).

*Relation roles* permit position independent naming of the arguments (elements) of a *relation*.

Note: This is similar to the distinction between positional and named arguments to procedures in programming languages.

For symmetric *binary relations* we reuse *relation roles* to indicate multiple arguments (*link ends*), since the arguments (*link ends*) are to be treated identically.

*Relation_Role* shall participate in the following association:

— *relation_role_set* (**Error! Reference source not found.**) which specifies the set of *Relation_Roles* belonging to a *Relation.*

*Relation_Role* may participate in the following association:

— *Link_End* (8.1.4.1) which identifies the role that a *Concept* (8.1.2.1) plays in a *Link* (8.1.2.6).

The *Relation_Role* class has two attributes: *multiplicity* and *ordinal*.

#### 8.1.2.5.2 Attributes of Relation_Role

##### 8.1.2.5.2.1 multiplicity

| | |
|---|---|
| Attribute name: | ***multiplicity*** |
| Definition: | number of *links* which must (logically) be members of the source *relation* of this role, differing only by an *end* with this *role* as an *end_role*.<br><br>For example, if a relation *purchase* with an arity of 3 has roles *buyer*, *seller*, and *item*, then a multiplicity of 0..1 on the *buyer* role means that it is not permitted for more than one member of the *purchase* relation to involve both the same *seller* and the same *item* (differing only in the *buyer*). |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Natural_Range* (5.1.8) |

##### 8.1.2.5.2.2 ordinal

| | |
|---|---|
| Attribute name: | ***ordinal*** |
| Definition: | order of the *relation role* among other *relation roles* in the *relation*. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Integer* (5.1.6) |
| Comment: | *ordinal* allows the ordering of the *Concepts,* represented in the *Relation* by the *Relation_Roles,* to be specified. This may be necessary if the ordering of the *Concepts* changes the meaning of the *Relation*. |

—— End of attributes of *Relation_Role* ——

#### 8.1.2.6 Link

A *link* is a member of a *relation*. In relational database parlance, a *link* would be a tuple (row) in a *relation* (table). *Link* is a class that represents a *link*. *Link* is a subtype of *Assertion* (8.1.2.3), and as such is included in one or more *Concept_Systems* (8.1.2.2) through the *assertion_inclusion* (8.1.3.5) association.

*Link* shall participate in the following associations:

— *assertion_inclusion* (8.1.3.5) with one or more *Concept_Systems* in which it is included.

*Link* may participate in the following associations:

— *relation_membership* (8.1.3.7) with exactly one *Relation* of which it is a *member*.

— *Link_End* (8.1.4.1) with two or more *Concepts* (8.1.2.1) and one or more *Relation_Roles* (8.1.2.5).

Note: a symmetric *binary relation* may use the same *relation role* for both *link ends*.

*Link* has no attributes.

### 8.1.3 Associations of the Concept_System region

#### 8.1.3.1 concept_system_membership

The *concept_system_membership* association specifies the inclusion of zero or more *Concepts* (8.1.2.1) in one or more *Concept_Systems* (8.1.2.2).

*concept_system_membership* has two roles:

— *including_concept_system* (verb form: *is_included_in*) which references a *Concept_System;*

— *member_concept* (verb form: *has_member_concept*) which references a *Concept*.

Each *Concept* shall have a *concept_system_membership* association with at least one *Concept_System*.

#### 8.1.3.2 concept_source

The *concept_source* association specifies the *Concept_System* (8.1.2.2) that is the source of a *Concept* (8.1.2.1).

*concept_source* has two roles:

— *source* (verb form: *has_source*) which references a C*oncept_System;*

— *specified_concept* (verb form: *specifies_concept*) which references a *Concept*.

Each *Concept* .shall have exactly one *Concept_System* specified as its source.

The *source Concept_System* establishes an explicit minimum scope within which the identity of the *Concept* can be taken to have been determined, in the sense that there should be no other *Concept* within that same scope which represents the same meaning. In some registries (including presumably all Edition 2 implementations), this scope may always be the registry *Concept_System*, thus making explicit the expectation that there shall always be at most one *Concept* within that entire registry representing any given meaning. In other registries the scope may generally be much narrower, reflecting a lack of determination having (necessarily) been made as to whether the same meaning may or may not also be represented by one or more other *Concepts* in the registry, from a different source(s).

The source *Concept_System* also provides a basis for capturing the set of assertions pertaining to that *Concept* (generally including many *Assertions* (8.1.2.3) that the *Concept* does not participate in directly) which are to be taken as committed to when referencing that *Concept.* In some registries in which all *Concepts* have a distinguished registry *Concept_System* as their source, all *Assertions* may also be included in that same registry *Concept_System*, thus making explicit a uniform ontological commitment spanning the entire registry. In other registries, all *Concepts* may have the registry as their source, but discrimination may be made between *Assertions* included in that registry *Concept_System*, and other *Assertions* which are registered but not ontologically committed to by reference to *Concepts* in the registry. In yet other registries there may be many different *Concept* instances representing either similar or even arguably identical meanings, but with some potentially critical difference(s) in semantics represented by the distinct sets of ontological commitments (*Assertions*) included in their respective source *Concept_Systems*.

#### 8.1.3.3 concept_system_reference

The *concept_system_reference* association specifies the reference of zero or more referenced *Concept_Systems* (8.1.2.2) by zero or more referencing *Concept_System*s.

*concept_system_reference* has two roles, both of which reference instances of the class *Concept_System*:

— *referenced_concept_system* (verb form: *has_ referenced_concept_system*);

— *referencing_concept_system* (verb form: *has_ referencing_concept_system*).

A *referenced_concept_system* may be referenced by zero or more *referencing_concept_systems*. A *referencing_concept_system* may reference zero or more *referenced_concept_systems*.

#### 8.1.3.4 concept_system_importation

The *concept_system_importation* association specifies the importation of zero or more imported *Concept_Systems* (8.1.2.2) by zero or more importing *Concept_System*s. Such importation specifies that all *Concepts* () and *Assertions* () included in the imported Concept_System are also to be included in the importing Concept_System.

*concept_system_importation* has two roles, both of which reference instances of the class *Concept_System*:

— *imported_concept_system* (verb form: *has_ imported_concept_system*);

— *importing_concept_system* (verb form: *has_ importing_concept_system*).

An *imported_concept_system* may be imported by zero or more *importing_concept_system*s.

An *importing_concept_system* may import zero or more *imported_concept_system*s.

#### 8.1.3.5 assertion_inclusion

The *assertion_inclusion* association specifies the inclusion of an *Assertion* (8.1.2.3) in one or more *Concept_Systems* (8.1.2.2).

*assertion_inclusion* has two roles:

— *assertor* (verb form: *asserted_by*) which references a C*oncept_System;*

— *included_assertion* (verb form: *includes*) which references an *Assertion*.

An *included_assertion* shall be *asserted_by* one or more *Concept_Systems.*

An *assertor Concept_System* may include zero or more *Assertions*.

#### 8.1.3.6 assertion_term

The *assertion_term* association specifies the use of one or more *Concepts* (8.1.2.1) as *terms* in zero or more *Assertions* (8.1.2.3).

*assertion_term* has two roles:

— *term* (verb form: *uses_term*) which references a C*oncept;*

— *assertion* (verb form: *used_by*) which references an *Assertion*.

An *assertion* shall use one or more *terms.* A *term* may be used by zero or more *assertions*.

#### 8.1.3.7 relation_membership

The *relation_membership* association specifies the membership of zero or more *Links* (8.1.2.6) as *members* in exactly one *Relation* (8.1.2.4).

*relation_membership* has two roles:

— *relation* (verb form: *member of*) which references a *Relation*;

— *member* (verb form: *has member*) which references a *Link*.

A *member* shall be a member of exactly one *relation.* A *relation* may have zero or more *members*.

#### 8.1.3.8 relation_role_set

The *relation_role_set* association specifies the *Relation_Roles* (8.1.2.5) that participate in the *Relation* (8.1.2.4).

*relation_role_set* has two roles:

— *role* (verb form: *has_role*) which references a *Relation_Role*;

— *source* (verb form: *has_source*) which references a *Relation*.

A *role* shall have exactly one *source.* A s*ource* shall have one or more *roles*.

The *relation_role_set* association is a strong containment relation. Hence, if a Relation is deleted all of its roles (Relation_Roles) are also deleted.

#### 8.1.3.9 link_end_role

The *link_end_role* association specifies the *Relation_Role* (8.1.2.5) that is the *role* for a *Link_End* (8.1.4.1).

*relation_role_set* has two roles:

— *role* (verb form: *has_role*) which references a *Relation_Role*;

— *source* (verb form: *has_source*) which references a *Relation*.

A *role* shall have exactly one *source.* A s*ource* shall have one or more *roles*.

The *relation_role_set* association is a strong containment relation. Hence, if a Relation is deleted all of its roles (Relation_Roles) are also deleted.

### 8.1.4 Association Classes in the Concept_System Region

#### 8.1.4.1 Link_End Association Class

The *Link_End* association class models the association between *Links* (8.1.2.6) and *Concepts* (8.1.2.1) (ends). This is used to represent the relationship between an n-tuple (row) of a relation and the values for the fields (arguments) of the n-tuple. Hence, a *link_end* association is used to model the instantiation of a *Relation_Role* (8.1.2.5) for a particular *Link* (tuple, row) of a *Relation* (8.1.2.4).

The *Link_End* association class has two roles:

— link (verb form: has_link) which references a *Link*;

— end (verb form: has_end) which references a *Concept*.

An end (*Concept*) may have zero or more links (*Links*). A link (*Link*) must have at least two, and possibly more ends (*Concepts*).

Finally, a *Link_End* association class has a *link_end_role* association () to the *Relation_Role* which the end (*Concept*) is intended to fulfil within a *Link*.

**Constraint:** It must be the case that every role (*Relation_Role*) specified in the *Link_End* association must correspond to a *Relation_Role* which is a role of the *Relation* of the *Link* to which the *Link_End* is associated.

## 8.2 Classification metamodel region

### 8.2.1 Overview

EDITOR'S NOTE #30. (Action Required) Issue 57 identifies the need to be able to classify an Administered_Item differently in different contexts. This issue has not been addressed.

The Classification region is illustrated in Figure 8-2. The purpose of this region is to provide a facility to use *Concept_Systems* (8.1.2.2) to model *classification schemes*.

*Classification schemes* are intended to permit the classification of arbitrary objects into hierarchies (or partial orders), whereas *concept systems* are used to enumerate and possibly classify *concepts*. However, since the structures are similar, we use the *Concept_System* structures of the metamodel to model *classification schemes* as well.

*Concept_Systems* may be used as *classification schemes* to classify *Classifiable_Items* within a registry, but some *classification schemes* will be more applicable to classifying objects in the real world than items in a registry. If the objects to be classified are not in the registry, the classification scheme may still be recorded using the *Concept_System* structures.

A *classification scheme* may be a taxonomy, a network, an ontology, or any other terminological system. The classification may also be just a list of controlled vocabulary of property words (or terms). The list might be taken from the "leaf level" of a taxonomy.

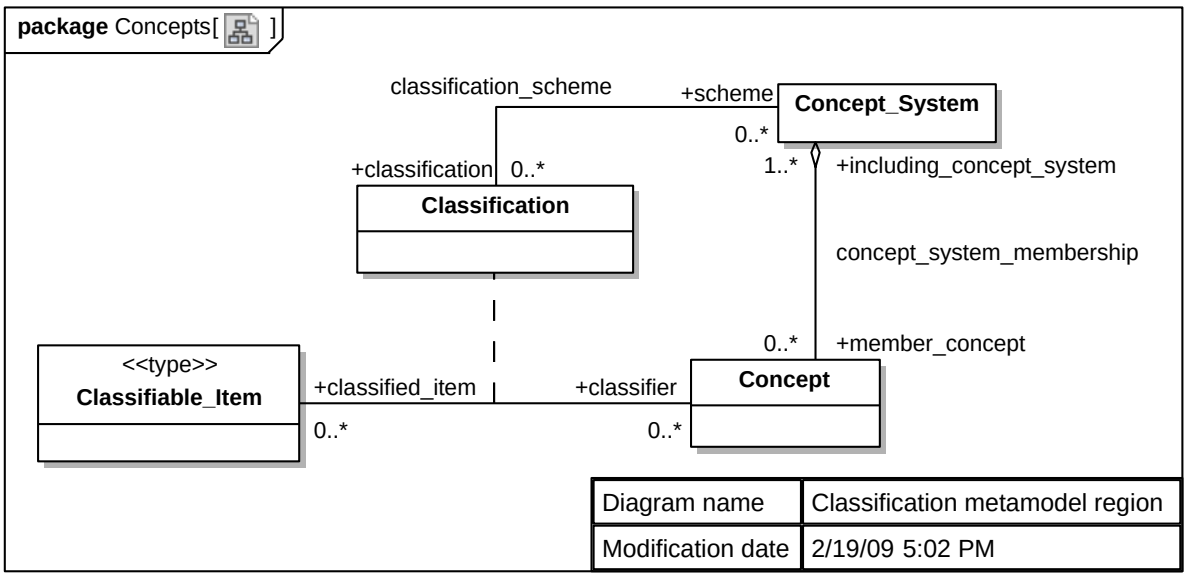



**Figure 8-2 — Classification metamodel region**

### 8.2.2 Classes in the Classification region

#### 8.2.2.1 Classifiable_Item

*Classifiable_Item* is an abstract supertype of all classes which might be classified (organized into a hierarchical structure or partial order).

A *Classifiable_Item* may be classified in zero or more *classification_schemes*, by associating it with one or more *classifier Concepts* as represented by the *Classification* association class in Figure 8-2*.* Such classification is optional.

#### 8.2.2.2 Concept_System

*Concept_System* is specified in 8.1.2.2. *Concept_System* is used to model a *classification scheme* by defining the nodes of the *classification scheme* as *Concepts* (8.1.2.1) in the *Concept_System*. Hierarchical ordering or other relationships among the nodes of the *classification scheme* can be specified using *Relations* (8.1.2.4) among the *Concepts*. The *Relations* may be named and defined by making them *Designatable_Items* (6.2.2.2)*.*

**Constraint:** *Concept_Systems* used as *classification schemes* shall be constrained to be partial orders, i.e., the *Relation*s among the *Concepts* must not contain any cycles. Partial orders may be represented as directed acyclic graphs (DAGs). However, the *Concept_System* need not be restricted to a hierarchy (i.e., a tree). The restriction of *classification schemes* to be partial orders is commonplace in the terminology and ontology communities.

#### 8.2.2.3 Concept

*Concept* is specified in 8.1.2.1. For purposes of modelling a *classification scheme*, *Concept* is used to represent a node in the *classification scheme.* The *Concept* represents a partition of the *classification scheme* that is homogeneous with respect some characteristic.

### 8.2.3 Associations Classes in the Classification Region

#### 8.2.3.1 Classification association class

The *Classification* association class is used to record the classification of a *Classifiable_Item* into a group designated by a *Concept* (8.1.2.1) in a *Concept_System* (8.1.2.2).

A *Classification* association has two roles:

— classified_item (verb form: has_classified_item) which references an instance of *Classified_Item;*

— classifier (verb form: classified_as) which references an instance of *Concept*.

The exact semantics of the *Classification* association are not specified by this standard, but will depend upon way in which the *classification scheme* is used. For example, the *Classification* association might signify either an "is-a" or an "instance-of" relationship.

A *Classifiable_Item* may be classified by zero or more *Concepts*. A *Concept* may classify zero or more *Classifiable_Items*.

### 8.2.4 Associations in the Classification Region

#### 8.2.4.1 classification_scheme

*classification_scheme* associates a *Classification* association class with zero or more *Concept_Systems* within which the classification occurs*.*

The association has two roles:

— classification (verb form: has_classification ) which references zero or more *Classifications*

— scheme (verb form: has_scheme ) which references zero or more *Concept_Systems*.

A *scheme* may have zero or more *Classifications*. A *Classification* may have zero or more *schemes*.

**Constraint:** It must be the case that the *Concept_System(s)* associated with the *Classification* through the *classification_scheme* association are the same *Concept_System(s)* associated with the *Concept(s)* that participate in the *Classification*.

#### 8.2.4.2 concept_system_membership

The *concept_system_membership* association is described in (8.1.3.1).

# 9 Binary_Relations Package

## 9.1 Binary_Relations metamodel region

### 9.1.1 Overview

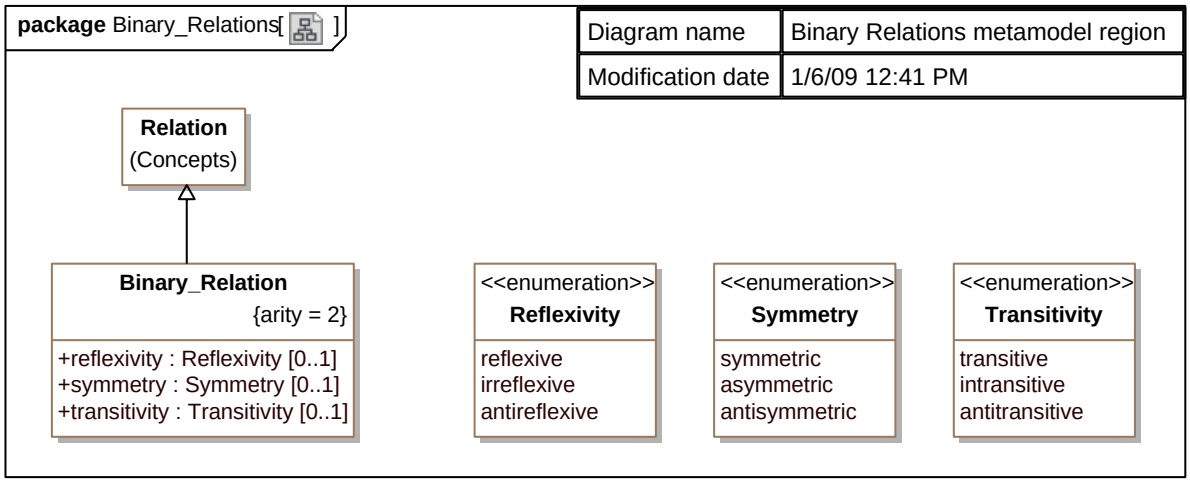



**Figure 9-1 — Binary Relations metamodel region**

### 9.1.2 Classes in the Binary_Relations metamodel region

#### 9.1.2.1 Relation

The *Relation* class is described in 8.1.2.4.

#### 9.1.2.2 Binary_Relation

##### 9.1.2.2.1 Description of Binary_Relation

A *binary relation* is a *relation* of arity 2 (i.e. having two *link ends*). *Binary_Relation* is a subclass of *Relation* (8.1.2.4) used to model *binary relations*.

Most common semantic relations are binary, e.g., equals, less than, greater than, is-a, part-of, etc. A example of a relation which is not binary would be betweeness. Binary relations are commonly represented as edges (or directed edges for asymmetric binary relations) in graphs, cf. the RDF (Resource Description Framework) of the W3C.

Below is a table of examples of some binary relationships and their characterization.

| Relation | Symmetry | Reflexivity | Transitivity |
|---|---|---|---|
| equals | symmetric | reflexive | transitive |
| not equals | symmetric | antireflexive | intransitive |
| less than | antisymmetric | antireflexive | transitive |
| less than or equal | asymmetric | reflexive | transitive |
| similar | symmetric | reflexive | intransitive |

**Table 9-1: Examples of binary relations and their characterization**

The *Binary_Relation* class has three enumeration attributes: reflexivity, symmetry, and transitivity.

#### 9.1.2.2.2 Attributes of Binary_Relation

##### 9.1.2.2.2.1 reflexivity

Attribute name: ***reflexivity***

Definition: characterization of the *Binary_Relation* as: reflexive, irreflexive or antireflexive.

Obligation: Optional

Multiplicity: *0..1*

Data type: *Reflexivity* (9.1.2.3)

Notes: A *Binary_Relation*, R, is reflexive if for all x, R(x,x) is true. Equality is an example of a reflexive relation.

A *Binary_Relation*, R, is irreflexive if it is not reflexive. i.e., R(x,x) is not necessarily true for all x.

A *Binary_Relation*, R, is antireflexive if for all x, R(x,x) is false. Inequality is an example of an antireflexive relation.

An antireflexive relation is also irreflexive, but antireflexive is a more specific characterization.

##### 9.1.2.2.2.2 symmetry

Attribute name: ***symmetry***

Definition: characterization of the *Binary_Relation* as: symmetric, asymmetric or antisymmetric

Obligation: Optional

Multiplicity: *0..1*

Data type: *Symmetry* (9.1.2.4)

Notes: A *Binary_Relation*, R, is symmetric if for all x, y: R(x,y) implies R(y,x). Examples of symmetric relations are ‘equals’, ‘not equals’, ‘within-2-miles-of’, etc. Note that symmetry does not imply reflexivity. For example, the inequality relation is symmetric, but antireflexive.

A *Binary_Relation*, R, is asymmetric if for all x,y: R(x,y) does not imply R(y,x). In terms of this metamodel, asymmetric *Relations* have two distinguishable (non-identical) roles, one for each *end* of each *Link*. Examples of asymmetric relations include: less than, likes, father of, etc.

A *Binary_Relation*, R, is anti-symmetric if for all x,y: R(x,y) implies not R(y,x). 'Less than' is an example of an antisymmetric relation.

Note: An antisymmetric relation is also asymmetric, but antisymmetric is a more specific characterization. An asymmetric relation is not necessarily antisymmetric (consider less than or equals).

#### 9.1.2.2.2.3 transitivity

| | |
|---|---|
| Attribute name: | ***transitivity*** |
| Definition: | characterization of the *Binary_Relation* as: transitive, intransitive or antitransitive |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Transitivity* (9.1.2.5) |
| Notes: | A Binary_Relation, R, is transitive, if for all x,y,z: R(x,y) and R(y,z) implies R(x,z). Examples of transitive relations include equality, less than, and less than or equals.<br><br>A Binary_Relation, R, is intransitive if it is not transitive i.e., R(x,y) and R(y,z) does not imply R(x,z).<br><br>A Binary_Relation, R, is antitransitive if for all x,y,z: R(x,y) and R(y,z) implies not R(x,z).<br><br>Note: An antitransitive relation is also intransitive, but antitransitive is a more specific characterization. |

—— End of attributes of Binary_*Relation* ——

### 9.1.2.3 Reflexivity

*Reflexivity* is an enumeration of the values: *reflexive, irreflexive, antireflexive*. *Reflexivity* is used as the datatype of the *reflexivity* attribute of *Binary_Relation.*

### 9.1.2.4 Symmetry

*Symmetry* is an enumeration of the values: *symmetric, asymmetric, antisymmetric*. *Symmetry* is used as the datatype of the *symmetry* attribute of *Binary_Relation.*

### 9.1.2.5 Transitivity

*Transitivity* is an enumeration of the values: *transitive, intransitive, antitransitive*. *Transitivity* is used as the datatype of the *transitivity* attribute of *Binary_Relation.*

# 10 Data Description Package

EDITOR'S NOTE #31. (Action required) This clause needs some rework to align the formatting with that now used in the preceding clauses. Some of the CD1 ballot comments on the descriptions of associations still need to be applied.

## 10.1 High-level Data Description metamodel

### 10.1.1 Overview

A high level overview of the metamodel can be found in Figure 10-1. It shows four classes: *Conceptual_Domain* (10.1.2.2), *Value_Domain* (10.1.2.3), and *Data_Element* (10.1.2.4) and *Data_Element_Concept* (10.1.2.5). The Figure also shows four associations among the four classes: *value_domain_meaning* (10.1.3.1), *data_element_domain* (10.1.3.2), *data_element_meaning* (10.1.3.3), *data_element_concept_domain* (10.1.3.4).

The following text describes the classes and associations shown in the Figure. It also describes a constraint on the high level metamodel not visible in the UML diagram. More detailed descriptions, e.g., of the class attributes, follow in subsequent subclauses.

Figure 10-1 — High-level Data Description metamodel can be partitioned into two horizontal parts, one upper part comprised of *Data_Element_Concept* and *Conceptual_Domain* and a second lower part comprised of *Data_Element* and *Value_Domain*. This view effectively splits the metamodel between a conceptual (or semantic) level (at the top) and a representational level (below). The representational level describes the information artifacts (in contrast to the semantic constructs of the upper level).

This high-level metamodel omits many details, e.g. attributes and some associations, in the interest of clarity of exposition. For a complete characterization of the metamodel the reader must consult the more detailed discussions which follow.

EDITOR'S NOTE #32. (Action required) Issue 107 calls for improved support of Complex Data Types.

EDITOR'S NOTE #33. (Action required) Issue 111 calls for the addition of support for data groups such as database tables and record definitions. Such support should also consider inclusion, encapsulation, stereotyping and inheritance.

EDITOR'S NOTE #34. (Action required) Should we add support for XML documents and/or XML schemas?

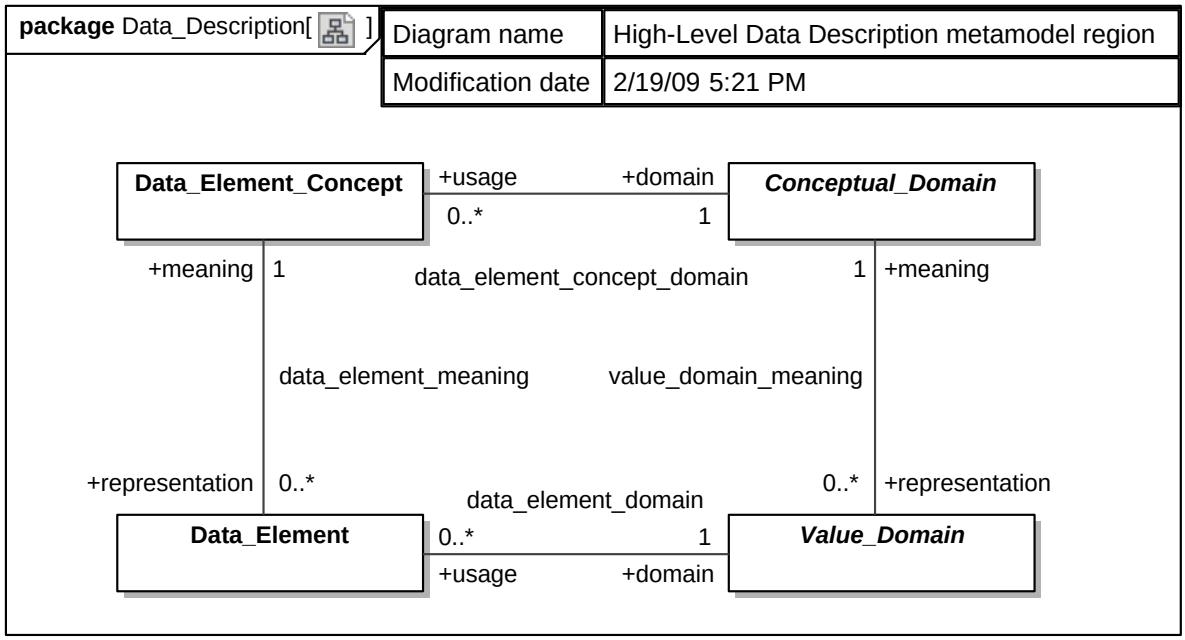



**Figure 10-1 — High-level Data Description metamodel**

### 10.1.2 Classes of High-level Data Description Metamodel

#### 10.1.2.1 Overview

The classes shown in Figure 10-1 are described below starting with *Conceptual_Domain* (10.1.2.2), and proceeding clock-wise around the Figure.

#### 10.1.2.2 Conceptual_Domain class

*Conceptual_Domain* is a class that represents a *conceptual domain*. A *conceptual domain* is a set of *value meanings*, which may either be enumerated or expressed via a description.

For example, one possible *conceptual domain* could be countries of the world. It might be associated with two *value domains*: three letter country codes, and full country names. The *conceptual domain* might be used in several *data element concepts*, e.g., "person's country of residence", "person's country of birth", "person's country of citizenship".

A *conceptual domain* can facilitate the mapping of equivalent values of two or more *value domains* that share the *conceptual domain*.

*Conceptual_Domain* is a class with two associations:

— *data_element_concept_domain* (to *Data_Element_Concept);*

— *value_domain_meaning* (to *Value_Domain)*.

*Conceptual_Domain* is further described in 10.3.2.1.

#### 10.1.2.3 Value_Domain class

*Value_Domain* is a class that represents a *value domain*. A *value domain* is a collection of *permissible values*. It provides representation, but has no implication as to what *data element concept* the values are associated

with, nor what the values mean. *Permissible values* are designations, bindings of signs (values) to their corresponding *value meanings*.

*Value_Domain* is associated with a *Conceptual_Domain* through the association *value_domain_meaning*. Through this association, a *value domain* provides a representation for the *conceptual domain* and the *conceptual domain* provides meaning for the *value domain*.

An example of a *conceptual domain* and a set of *value domains* is ISO 3166, Codes for the representation of names of countries. For instance, ISO 3166 describes the set of seven *value domains*: short name in English, official name in English, short name in French, official name in French, alpha-2 code, alpha-3 code, and numeric code.

Additional examples of *value domains* would be:

— Sex which contains two designations (*permissible values*), M -> Male and F-> Female, and

— Parent which contains two designations (*permissible values*), M -> Mother and F -> Father.

Note that these two *value domains* are defined over the same set of values (signs), but they are mapped to separate *conceptual domains*.

*Value_Domain* is a class with two associations:

— *value_domain_meaning* (to *Conceptual_Domain*) (which is described above);

— *data_element_domain* (to *Data_Element*) (which is described below).

*Value_Domain* is further described in 10.3.2.5.

*Value domains* may be reused for multiple *data elements*, see the discussion of countries of the world in 10.1.2.2 above.

#### 10.1.2.4 Data_Element class

*Data_Element* is a class that represents a *data element*. A *data element* is considered to be a basic unit of data of interest to an organization. It is a unit of data for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. Examples of *data element* include: a column in a table of a relational database, a field in a record or form, an XML element, the attribute of a Java class, or a variable in a program. The description of *data elements* is a major purpose of ISO/IEC 11179 Metadata Registries.

*Data_Element* is a class with two associations:

— *data_element_meaning* (to *Data_Element_Concept*);

— *data_element_domain* (to *Value_Domain*).

*Data_Element* is further described in 10.5.2.1.

#### 10.1.2.5 Data_Element_Concept class

*Data_Element_Concept* is a class that represents a *data element concept*. A *data element concept* is a concept that can be represented in the form of a *data element*, described independently of any particular representation.

A *data element concept* is a usage of a *conceptual domain*, e.g., “person’s country of residence” vs. “country”, which effectively narrows the meaning of the *conceptual domain*.

A *data element concept* is an abstraction of one or more *data elements*. Each *data element* addresses issues of concrete representation, e.g., codes, measurement units, etc. A *data element concept* may be represented by multiple *data elements*, which may vary in their *value domains*.

A *data element concept* can facilitate the mapping of equivalent values of two or more *data elements* that share the *data element concept*.

*Data_Element_Concept* is a class with two associations:

— *data_element_concept_domain* (to *Conceptual_Domain*);

— *data_element_meaning* (to *Data_Element*).

*Data_Element_Concept* is further described in 10.2.2.3.

### 10.1.3 Associations of the High Level Metamodel

#### 10.1.3.1 value_domain_meaning Association

The association *value_domain_meaning* binds a *Value_Domain* to a *Conceptual_Domain*. The association has two roles:

— *meaning* (verb form: gives meaning to)

— *representation* (verb form: provides representation for).

The *meaning* role specifies the *conceptual domain* of a *value domain*. The *representation* role specifies the *value domain*(s) of a *conceptual domain*. Each *meaning* (conceptual domain) may have zero or more associated *representations* (*value domains*). Each *representation* (*value domain*) has exactly one associated *meaning* (*conceptual domain*).

A *value domain* is a collection of *permissible values* which are *designations*, the mappings between *value meanings* and *values* (*signs*). Note that the existence of a *value_domain_meaning* association between a *Conceptual_Domain* and a *Value_Domain* implies the existence of associations between the corresponding individual *value meanings* and *values*. These associations (*designations*) are recorded as *Permissible_Values* in this metamodel).

Note that in this metamodel, *Value_Domains* are constrained to have a unique set of *meanings* (the associated *Conceptual_Domain*), i.e., a *Value_Domain* is a function from *Values* to *Value_Meanings*. If for some reason one wanted to reuse a *Value_Domain* (and the associated values, e.g., a code set) for more than one meaning (e.g. see the additional examples in 10.1.2.3), one is forced to create another *Value_Domain* and another set of *Permissible_Values*. This constraint is enforced so that within a *Value_Domain* one can unambiguously determine the value meanings (in the *Conceptual_Domain*) for the values (in the *Value_Domain*) associated with a *Data_Element*. (See discussion under Constraints in Section 10.1.4.1)

#### 10.1.3.2 data_element_domain Association

The *data_element_domain* association binds a *Data_Element* to a *Value_Domain* which specifies the *values* which may be stored in the *data element*. The association has two roles:

— *usage* (verb form: uses), which specifies a *Data_Element* which *uses* a *Value_Domain*;

— *domain* (verb form: provides_values_for), which specifies a *Value_Domain provides values for* the *Data_Element*.

A *usage* (*data element*) has exactly one *domain* (*value domain*). A *domain* (*value domain*) may have zero or more *usages* (*data elements*)

#### 10.1.3.3 data_element_meaning Association

The *data_element_meaning* association binds a *Data_Element* to its *Data_Element_Concept*. The association has two roles:

— meaning (verb form: provides meaning for), which specifies the *Data_Element_Concept* which *provides meaning for* a *Data_Element*;

— representation (verb form: represents), which specifies a *Data_Element* which *represents* the *Data_Element_Concept*.

Each *representation* (*data element*) has exactly one *meaning* (*data element concept*). However, a *meaning* (*data element concept*) may have zero or more *representations* (*data elements*).

#### 10.1.3.4 data_element_concept_domain Association

The *data_element_concept_domain* association binds a *Data_Element_Concept* to its *Conceptual_Domain*. The association has two roles:

— *usage* (verb form: uses), which specifies the *Data_Element_Concept* which *uses* a *Conceptual_Domain*;

— *domain* (verb form: provides_domain_for) which specifies the *Conceptual_Domain* which *provides the domain for* a Data_Element_Concept.

Each *usage* (*data element concept*) has exactly one *domain* (*conceptual domain*). Each *domain* (*conceptual domain*) may have zero or more associated *usages* (*data element concepts*).

The *data_element_concept_domain* association narrows the scope (*meaning*) of a *Conceptual_Domain* to that of the *Data_Element_Concept*, e.g., “person’s country of birth” (*data element concept*) vs. “country” (*conceptual domain*).

### 10.1.4 Constraints of the High Level Metamodel

#### 10.1.4.1 Equality of mappings from data element to conceptual domain

There are two paths in the metamodel from the *Data_Element* class to the *Conceptual_Domain* class. One can either proceed clockwise from *Data_Element* class via the *data_element_meaning* association to *Data_Element_Concept* class and then via *data_element_concept_domain* association to the *Conceptual_Domain* class; or alternatively, one can proceed counterclockwise from the *Data_Element* class via the *data_element_domain* association to the *Value_Domain* class and then via the *value_domain_meaning* association to the *Conceptual_Domain* class.

It must be the case, that if we start from a specific instance of *Data_Element* class, that we end at the same instance of the *Conceptual_Domain* class, regardless of whether we proceed clockwise or counterclockwise through the associations of the metamodel. This constraint is not visible in the UML model.

Formally, we assert that for every **x** such that **x** is a member of the *Data_Element* class, then the *domain* of the *meaning* of **x** must equal the *meaning* of the *domain* of **x**. Note that (unfortunately) *domain* and *meaning* are used here twice to refer to different roles (functions).

Note that the possible inverse constraint (starting from Conceptual_Domain) is not true, because the associations are not functions (uniquely valued) in the inverse directions.

## 10.2 Data Element Concept region

### 10.2.1 Overview

The **Data Element Concept** region is illustrated in Figure 10-2 — Data_Element_Concept metamodel region. The purpose of this region is to maintain the information on the *concepts* related to *data elements*. The metadata objects in this region concern semantics. *Concepts* are independent of any internal or external physical representation. The metadata objects in this region are: *Conceptual_Domains*, *Data_Element_Concepts*, *Object_Classes* and *Characteristics*. *Object_Classes* and *Characteristics* may be combined to form *Data_Element_Concepts*. All of these metadata objects are subtypes of *Concept* (see 10.7).

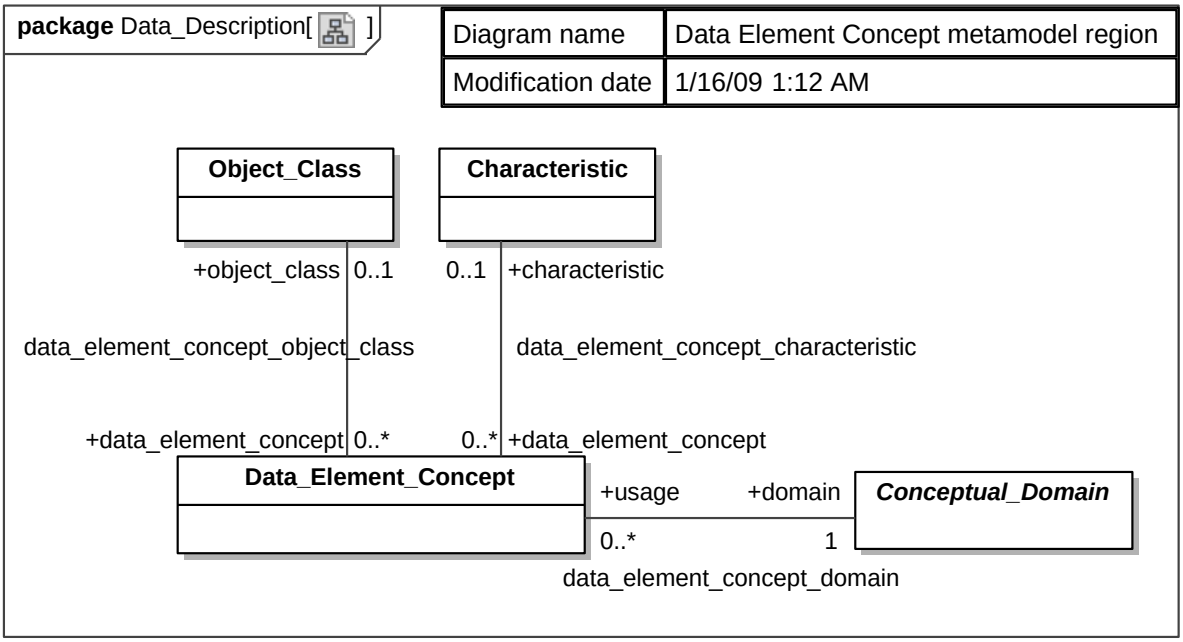



**Figure 10-2 — Data_Element_Concept metamodel region**

### 10.2.2 Classes in the Data_Element_Concept region

#### 10.2.2.1 Object_Class

*Object_Class* is a class which represents an *object class.* An *object class* is a *concept* that represents a set of ideas, abstractions, or things in the real world that can be identified with explicit boundaries and meaning and whose properties and behavior follow the same rules. It may be either a single or a group of associated concepts, abstractions, or things.

An *Object_Class* may have a *data_element_concept_object_class* association with zero or more *Data_Element_Concepts*, where the *object class* describes the ideas, abstractions or things in the real world that are represented by the *data element concept*.

Example: The *Object_Class* "Person" could be represented by the *Data_Element_Concept* "Person Country of Residence".

#### 10.2.2.2 Characteristic

EDITOR'S NOTE #35. (Action required) CD1 ballot comment JP04 points out the the definition of 'characteristic' (3.3.6) states that: *Characteristics are used for describing* ***concepts****.* The text below states that a 'characteristic is a concept'. If this is true, it should be reflected in its definition. If it is not true, this text

and corresponding text in 10.7 Types of Concepts in the Data Description Metamodel should be corrected.
Note: The above definition does not necessarily preclude a characteristic from also being a concept.

*Characteristic* is a class which represents a *characteristic*. A *characteristic* is a *concept* that represents an abstraction of a property of an object or set of objects. A *characteristic* is common to all of the members of a given *object class*. It may be any feature that humans naturally use to distinguish one individual object from another. It is the human perception of a single *characteristic* of an *object class* in the real world. It is conceptual and thus has no particular associated means of representation by which the *characteristic* can be communicated.

A *Characteristic* may have a *data_element_concept_characteristic* association with zero or more *Data_Element_Concepts*.

#### 10.2.2.3 Data_Element_Concept

*Data_Element_Concept* is a class which represents a *data element concept.* A *data element concept* is a specification of a *Concept* independent of any particular representation. A *data element concept* can be represented in the form of a *data element*.

A *Data_Element_Concept* may have a *data_element_concept_object_class* association with zero or one *Object_Class* and a *data_element_concept_characteristic* association with zero or one *Characteristic*.

The union of a *characteristic* and an *object class* provides significance beyond either that of the *characteristic* or the *object class*. A *data element concept* thus has a *definition* independent from the *definition* of the *object class* or the *characteristic.*

Every *Data_Element_Concept* must have exactly one *data_element_concept_domain* association with a *Conceptual_Domain* (10.3.2.1), where the *Data_Element_Concept* supplies a *usage* for the associated *Conceptual_Domain*.

Example: An association between the *Data_Element_Concept* “Person Country of Residence” and the *Conceptual_Domain* “Country”.

#### 10.2.2.4 Concept_Domain

*Conceptual_Domain* is described in 10.3.2.1 as part of the Conceptual and Value Domain region.

### 10.2.3 Associations in the Data_Element_Concept region

#### 10.2.3.1 data_element_concept_characteristic

*data_element_concept_characteristic* is an association between a *Data_Element_Concept* and a *Characteristic* that provides a criterion for the subdivision of a *Conceptual_Domain*. *Data element concept characteristic* has two roles: *criterion* (verb form: *has criterion*) and *subdivision* (verb form: *subdivides*). The *criterion* role references a *Characteristic* and the *subdivision* role references a *Data_Element_Concept*. A *Data_Element_Concept* may be associated with zero or one *criterion Characteristics*. A *Characteristic* may be associated with zero or more *subdivision Data_Element_Concepts*.

#### 10.2.3.2 data_element_concept_domain

A *data_element_concept_domain* is an association denoting the *Conceptual_Domain* that provides the domain for a *Data_Element_Concept.* Every *Data_Element_Concept* must be associated with exactly one *Conceptual_Domain*. A *Conceptual_Domain* may be associated with zero, one or more *Data_Element_Concepts*.

#### 10.2.3.3 data_element_concept_object_class

*Data element concept object class* is an association between a **Data_Element_Concept** and an **Object_Class** that represent a particular set of ideas, abstractions, or things in the real world that whose

properties and behaviour follow the a set of rules as represented by the **Data_Element_Concept**. *Data element concept object class* has two roles: *represented* (verb form: *represents*) and *representation* (verb form: *represented by*). The *represented* role references an **Object_Class** and the *representation* role references a **Data_Element_Concept**. A **Data_Element_Concept** may be associated with zero or one *represented* **Object_Classes**. An **Object_Class** may be associated with zero or more *representation* **Data_Element_Concepts.**

## 10.3 Conceptual and Value_Domain region

### 10.3.1 Overview

This region of the metamodel addresses the administration of *Conceptual_Domains* and *Value_Domains*. These domains can be viewed as logical code sets and physical code sets. *Conceptual_Domains* support *Data_Element_Concepts* and *Value_Domains* support *Data_Elements*. The region is illustrated in Figure 10-3 — Conceptual and value domain metamodel region.

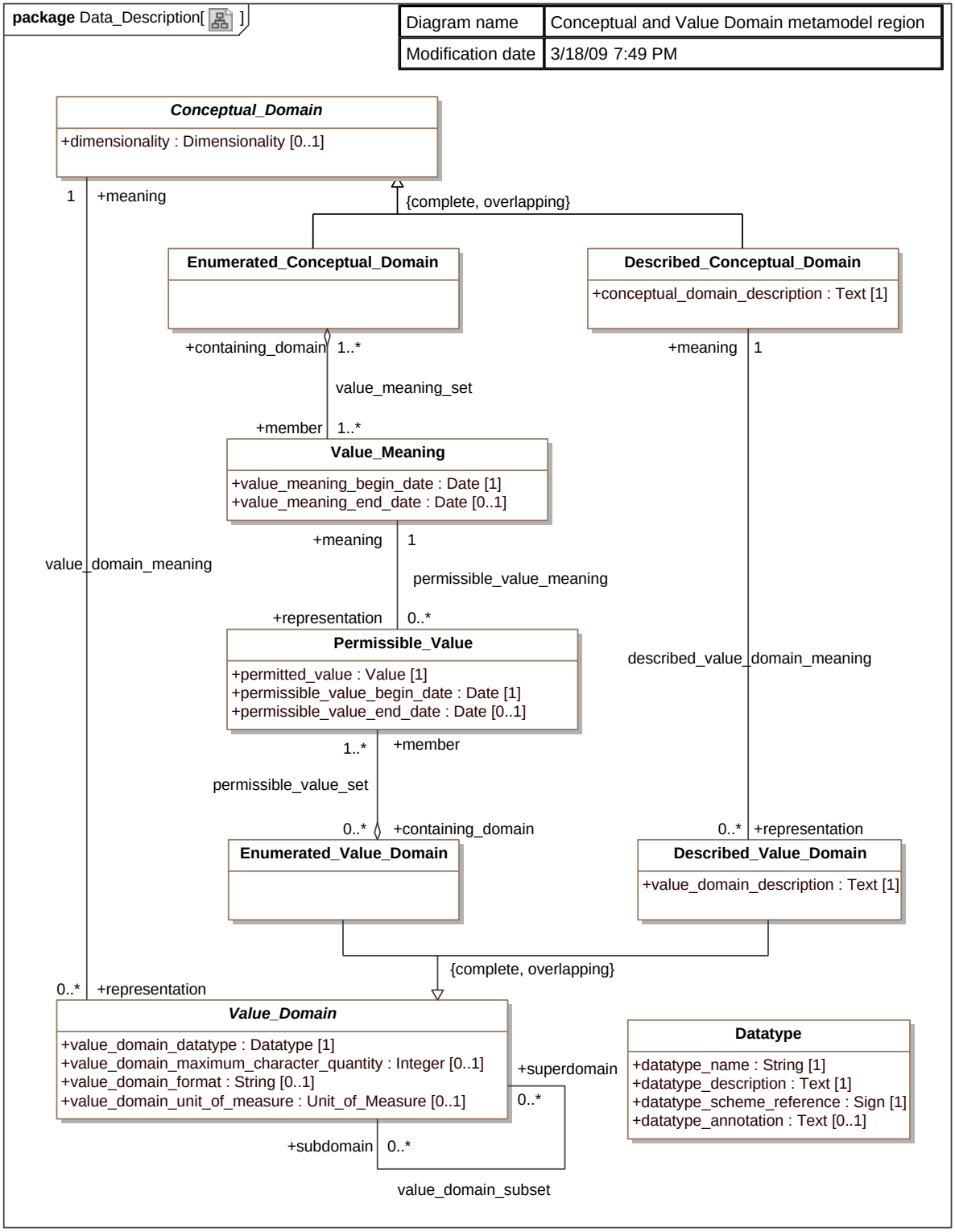



**Figure 10-3 — Conceptual and value domain metamodel region**

### 10.3.2 Classes in the Conceptual and Value_Domain region

*Conceptual_Domain*, *Dimensionalilty*, *Value_Meaning*, *Value_Domain* and *Unit_of_Measure* are each subtypes of *Registered_Item*, and hence of *Identified_Item* and *Designatable_Item* (see Figure 10). Such are the mechanisms by which they are identified and named, respectively. As subtypes of *Classifiable_Item*, they may also be classified within a Classification_Scheme.

#### 10.3.2.1 Conceptual_Domain

##### 10.3.2.1.1 Description of Conceptual_Domain

A *Conceptual_Domain* is a set of *Value_Meanings*, which may either be enumerated or expressed via a description. *Conceptual_Domain* is an abstract class, which has two possible subtypes: *Enumerated_Conceptual_Domain* and *Described Conceptual_Domain.* A *Conceptual_Domain* instance must be either or both an *Enumerated_Conceptual_Domain* or a *Described Conceptual_Domain. Conceptual_Domain* is a subtype of *Concept*.

NOTE In Figure 10-3, the use of *italics* in the name of *Conceptual_Domain* indicates that it is an abstract class.

The *Conceptual_Domain* class has one attribute, conceptual_domain_dimensionality, of type *Dimensionality*. The dimensionality attribute specifies the Dimensionality as elaborated in the discussion of the Dimensionality class below in Section

##### 10.3.2.1.2 Attributes of Conceptual_Domain

###### 10.3.2.1.2.1 conceptual_domain_dimensionality

| | |
|---|---|
| Attribute name: | ***conceptual_domain_dimensionality*** |
| Definition: | expression of measurement without units |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Dimensionality* (10.4.2.2) |
| Note 1: | When a *conceptual_domain_dimensionality* is specified, then any *Unit_of_Measure* (10.4.2.1) specified for any *Value_Domain* (10.3.2.5) that is based on this *Conceptual_Domain,* shall be consistent with this *dimensionality*. |
| Note 2: | ISO 31-0 specifies physical dimensions (e.g. length, mass, velocity). ISO/IEC 11179-3 also permits non-physical dimensions (e.g. value dimensions such as: currency, quality indicator) |

—— End of attributes of *Conceptual_Domain* ——

#### 10.3.2.2 Enumerated_Conceptual_Domain

A *Conceptual_Domain* sometimes contains a finite allowed inventory of notions that can be enumerated. Such a *Conceptual_Domain* is referred to as an *Enumerated_Conceptual_Domain.*

EXAMPLE: The notion of countries that is specified in ISO 3166, Codes for the representation of names of countries.

As a subtype of *Conceptual_Domain,* an *Enumerated_Conceptual_Domain* inherits the attributes and relationships of the former.

#### 10.3.2.3 Value_Meaning

##### 10.3.2.3.1 Description of Value_Meaning

Each member of an *Enumerated_Conceptual_Domain* has a *Value_Meaning* that provides its distinction from other members. In the example of ISO 3166, the notion of each country as specified would be the *Value_Meanings*. The representation of *Value_Meanings* in a registry shall be independent of (and shall not constrain) their representation in any corresponding *Value_Domain*. A particular *Value_Meaning* may have more than one means of representation by *Permissible_Values* — each from a distinct *Enumerated_Value_Domain. Value_Meaning* is a subtype of *Concept.*

*Value_Meanings* may participate in the *value_meaning_set* association and the *permissible_value_meaning* association. See discussion below.

The *Value_Meaning* class has two attributes: *value_meaning_begin_date*, and *value_meaning_end_date.*

The description of the *Value_Meaning* is specified by making the *Value_Meaning* a *Designatable_Item* (6.2.2.2) and using an associated *Definition* (6.2.2.4).

##### 10.3.2.3.2 Attributes of Value_Meaning

###### 10.3.2.3.2.1 value_meaning_begin_date

| | |
|---|---|
| Attribute name: | ***value_meaning_begin_date*** |
| Definition: | date at which this *Value_Meaning* became, or will become, a valid *Value_Meaning* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Date* (5.1.4) |
| Note: | A *registration authority* may determine whether this date is the date the *value meaning* becomes valid in a registry, or the date the *value meaning* becomes part of the source domain, or some other date. |

###### 10.3.2.3.2.2 value_meaning_end_date

| | |
|---|---|
| Attribute name: | ***value_meaning_end_date*** |
| Definition: | date on which the *Value_Meaning* ceased, or will cease, to be valid. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Date* (5.1.4) |
| Note 1: | The absence of the *value_meaning_end_date* indicates that the *Value_Meaning* is still valid. |
| Note 2: | A *registration authority* may determine whether this date is the date the *value meaning* becomes no longer valid in a registry, or the date the *value meaning* becomes no longer part of the source domain, or some other date. |

—— End of attributes of *Value_Meaning* ——

#### 10.3.2.4 Described_Conceptual_Domain

##### 10.3.2.4.1 Description of Described_Conceptual_Domain

A *Conceptual_Domain* that cannot be expressed as a finite set of *Value_Meanings* is called a *Described Conceptual_Domain*. It may be expressed via a description or specification, such as a rule, a procedure, or a

range (i.e., interval). As a subtype of *Conceptual_Domain,* a *Described Conceptual_Domain* inherits the attributes and relationships of the former.

The *Described Conceptual_Domain* class has one attribute: *described_conceptual_domain_description* which is of type *Text*. Each *Described Conceptual_Domain* class must have exactly one *described_conceptual_domain_description* attribute.

#### 10.3.2.4.2 Attributes of Described_Conceptual_Domain

##### 10.3.2.4.2.1 conceptual_domain_description

| | |
|---|---|
| Attribute name: | ***conceptual_domain_description*** |
| Definition: | description or specification of a rule, reference, or range for a set of all *Value_Meanings* for a *Conceptual_Domain.* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Text* (5.1.17) |

—— End of attributes of *Conceptual Domain* ——

### 10.3.2.5 Value_Domain

#### 10.3.2.5.1 Description of Value_Domain

One of the key components of a representation is the *Value_Domain*. A *Value_Domain* provides representation, but has no implication as to the *Data_Element_Concept* with which the values are associated, nor what the values mean.

A *Value_Domain* is an abstract class which is used to denote a collection of *Permissible_Values* associated with a *Conceptual_Domain*. A *Value_Domain* has two possible subtypes: an *Enumerated_Value_Domain* and a *Described_Value_Domain*. A *Value_Domain* must be either one or both an *Enumerated Valued* or a *Described_Value_Domain*.

NOTE In Figure 12, the use of *italics* in the name *Value_Domain* indicates that it is an abstract class.

A *Value_Domain* is associated with a *Conceptual_Domain*. A *Value_Domain* provides a representation for the *Conceptual_Domain*.

EXAMPLE: 'ISO 3166 Codes for the representation of names of countries' describes seven distinct *Value_Domains* for the single *Conceptual_Domain* 'names of countries'. The seven *Value_Domains* are: 'short name in English', 'official name in English', 'short name in French', 'official name in French', 'alpha-2 code', 'alpha-3 code' and 'numeric code'.

#### 10.3.2.5.2 Attributes of Value_Domain

##### 10.3.2.5.2.1 value_domain_datatype

| | |
|---|---|
| Attribute name: | ***value_domain_datatype*** |
| Definition: | *Datatype* used in a *Value_Domain* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Datatype* (10.3.2.9) |
| Note: | applies to <u>all</u> values in the *Value_Domain.* |

#### 10.3.2.5.2.2 value_domain_format

| | |
|---|---|
| Attribute name: | ***value_domain_format*** |
| Definition: | template for the structure of the presentation of the **value**(s)<br>EXAMPLE – YYYY-MM-DD for a date. |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *String* (5.1.16) |

#### 10.3.2.5.2.3 value_domain_maximum_character_quantity

| | |
|---|---|
| Attribute name: | ***value_domain_maximum_character_quantity*** |
| Definition: | maximum number of characters available to represent the *Data_Element* value |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Integer* (5.1.6) |
| Note: | Applicable only to character datatypes. |

#### 10.3.2.5.2.4 value_domain_unit_of_measure

| | |
|---|---|
| Attribute name: | ***value_domain_unit_of_measure*** |
| Definition: | *Unit_of_Measure* used in a *Value_Domain* |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Unit_of_Measure* (10.4.2.1) |
| Note: | applies to all values in the *Value_Domain*. |

—— End of attributes of *Value_Domain* ——

### 10.3.2.6 Enumerated_Value_Domain

An *Enumerated_Value_Domain* is one where the *Value_Domain* is expressed as an explicit set of two or more *Permissible_Values*. The *Enumerated_Value_Domain* class is a concrete subtype of the abstract class *Value_Domain*.

Each *Enumerated_Value_Domain* class shall participate in the *permissible_value_set* association.

### 10.3.2.7 Permissible_Value

#### 10.3.2.7.1 Description of Permissible_Value

A *Permissible_Value* is an expression of a *Value_Meaning* within an *Enumerated_Value_Domain*. It is one of a set of such values that comprises an *Enumerated_Value_Domain*.

Each *Permissible_Value* shall participate in the *permissible_value_meaning* association with exactly one *Value_Meaning*.

Each *Permissible_Value* may participate in the *permissible_value_set* association with zero or more *Enumerated_Value_Domains*.

#### 10.3.2.7.2 Attributes of Permissible_Value

##### 10.3.2.7.2.1 permitted_value

| | |
|---|---|
| Attribute name: | ***permitted_value*** |
| Definition: | the actual value of the *Permissible_Value* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Value* (5.1.18) |

##### 10.3.2.7.2.2 permissible_value_begin_date

| | |
|---|---|
| Attribute name: | ***permissible_value_begin_date*** |
| Definition: | date at which the *Permissible_Value* became valid |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | *Date* (5.1.4) |
| Note: | By imputation, this is also considered to be date at which the *Permissible_Value* was bound to the associated *Value_Meaning*, since the *permissible_value_meaning* association mandates that there must be exactly one meaning (*Value_Meaning*) for each representation (*Permissible_Value*). |

##### 10.3.2.7.2.3 permissible_value_end_date

| | |
|---|---|
| Attribute name: | ***permissible_value_end_date*** |
| Definition: | date at which the *Permissible_Value* ceased to be valid |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | *Date* (5.1.4) |
| Note 1: | By imputation, this is also considered to be date at which the *Permissible_Value* ceased to be bound to its associated meaning (*Value_Meaning*) via the *permissible_value_meaning* association. |
| Note 2: | The absence of the *permissible_value_end_date* attribute indicates that the *Permissible_Value* is still valid and (by imputation) still bound to its *Value_Meaning* via the *value_meaning* association. |

—— End of attributes of *Permissible_Value* ——

#### 10.3.2.8 Described_Value_Domain

##### 10.3.2.8.1 Description of Described_Value_Domain

A *Described_Value_Domain* is a concrete subtype of the abstract class *Value_Domain* which is characterized via a description or specification, such as a rule, a procedure, or a range (i.e., interval), rather than as an explicit set of *Permissible_Values*.. As a subtype of *Value_Domain,* a *Described_Value_Domain* inherits the attributes and relationships of the former.

##### 10.3.2.8.2 Attributes of Described_Value_Domain

##### 10.3.2.8.2.1 value_domain_description

| | |
|---|---|
| Attribute name: | ***value_domain_description*** |
| Definition: | description or specification of a rule, reference, or range for a set of all *Permissible_Values* for a *Value_Domain* |

Obligation: Mandatory

Multiplicity: *1*

Data type: *Text* (5.1.17)

—— End of attributes of *Described_Value_Domain* ——

#### 10.3.2.9 Datatype

##### 10.3.2.9.1 Description of Datatype

EDITOR'S NOTE #36. (Action required) In the following sentence, the text following 'for example' simply repeats the original statement but applied to a Data_Element, which is not so much an example, as how any value domain is supposed to be used. Some rewording would seem to be needed.

A *Value_Domain* is associated with a *Datatype* — a set of distinct values, characterized by properties of those values and by operations on those values, for example the category used for the collection of letters, digits, and/or symbols to depict values of a *Data_Element* determined by the operations that may be performed on the *Data_Element*.

##### 10.3.2.9.2 Attributes of Datatype

###### 10.3.2.9.2.1 datatype_name

Attribute name: ***datatype_name***

Definition: designation for the *Datatype*

Obligation: Mandatory

Multiplicity: *1*

Data type: String (5.1.16)

Note: The *datatype_name* is usually drawn from some external source, which in turn is designated by means of the mandatory *datatype_scheme_reference*.

###### 10.3.2.9.2.2 datatype_description

Attribute name: ***datatype_description***

Definition: descriptive information to further clarify the *Datatype*

Obligation: Mandatory

Multiplicity: *1*

Data type: Text (5.1.17)

###### 10.3.2.9.2.3 datatype_scheme_reference

Attribute name: ***datatype_scheme_reference***

Definition: reference identifying the source of the *Datatype* specification

Obligation: Mandatory

Multiplicity: *1*

Data type: Sign (5.1.15)

Note: In this edition of ISO/IEC 11179-3, the manner of reference is specified by the *registration authority*.

###### 10.3.2.9.2.4 datatype_annotation

Attribute name: ***datatype_annotation***

Definition: specifying information to further define the *Datatype*

| | |
|---|---|
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | Text (5.1.17) |

—— End of attributes of *Datatype* ——

### 10.3.3 Associations in the Conceptual and Value_Domain region

#### 10.3.3.1 value_domain_meaning Association

The *value_domain_meaning* association has two roles: meaning (verb form: means) and representation (verb form: represents). The meaning role refers to a *Conceptual_Domain* class. The representation role refers to a *Value_Domain* class. Each representation (*Value_Domain*) must have exactly one meaning (*Conceptual_Domain*). However, each meaning (*Conceptual_Domain*) may have zero or more representations (*Value_Domains*).

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the value_domain_meaning association.

#### 10.3.3.2 value_meaning_set Association

The *value_meaning_set* association has two roles: containing_domain (verb form: contains_domain) and member (verb form: has_member). The containing_domain role refers to an *Enumerated_Conceptual_Domain* class. The member role refers to a *Value_Meaning*. Each member (*Value_Meaning*) may have zero or more containing_domains (*Enumerated_Conceptual_Domain*). Each containing_domain (*Enumerated_Conceptual_Domain*) shall have one or more members (*Value_Meanings*). The value_meaning_set association is a weak containment association, which means that deletion of the containing *Enumerated_Conceptual_Domain* does not imply a cascading delete the contained *Value_Meanings*.

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the *value_meaning_set* association.

#### 10.3.3.3 described_value_domain_meaning Association

EDITOR'S NOTE #37. (Informational) It has been suggested that the described_value_domain_meaning association is redundant w.r.t. the value_domain_meaning association. However, the Editor notes that it does explicitly restrict the association to the two 'Described' subtypes. When a Conceptual_Domain and Value_Domain are a combination of Enumerated and Described domains, the two associations are not equivalent, though for navigation purposes they are still redundant. Should we include additional text explaining this?

The *described_value_domain_meaning* association has two roles: meaning (verb form: means) and representation (verb form: represents). The meaning role refers to a *Described_Conceptual_Domain* class. The representation role refers to a *Described_Value_Domain* class. Each representation (*Described_Value_Domain*) must have exactly one meaning (*Described_Conceptual_Domain*). However, each meaning (*Described Conceptual_Domain*) may have zero or more representations (*Described_Value_Domains*).

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the described_value_meaning association.

#### 10.3.3.4 permissible_value_meaning Association

The *permissible_value_meaning* association has two roles: meaning (verb form: means) and representation (verb form: represents). The meaning role refers to a *Value_Meaning* class. The representation role refers to a *Permissible_Value* class. Each representation (*Permissible_Value*) must have exactly one meaning

(*Value_Meaning*). However, each meaning (*Value_Meaning*) may have zero or more representations (*Permissible_Values*).

NOTE See discussion above under *Value_Meaning* for treatment of valid dates for *permissible_value_meaning* association. We impute valid dates for the *permissible_value_meaning* association from the *permissible_value_begin_date* and *permissible_value_end_date*.

#### 10.3.3.5 permissible_value_set Association

The *permissible_value_set* association has two roles: member (verb form: has member) and containing_domain (verb form: contains_domain). The member role refers to a *Permissible_Value* class. The contains_domain role refers to an *Enumerated_Value_Domain* class. Each member (*Permissible_Value*) may have zero or more containing_domains (*Enumerated_Value_Domains*). However, each containing_domain (*Enumerated_Value_Domain*) shall have one or more members (*Permissible_Values*). The *permissible_value_set* association is a weak containment relation, i.e., deletion of the containing domain does not cause a cascading delete of the members (*Permissible_Values*).

NOTE This version of the metamodel lacks any mechanism to specify the valid dates for the permissible_value_set association.

#### 10.3.3.6 value_domain_subset Association

### 10.3.4 Additional Constraints of the Conceptual and Value_Domain region

#### 10.3.4.1 Overview

This sub-clause specifies additional constraints that are not included in the UML diagram.

#### 10.3.4.2 value_domain_meaning Association Constraints

**Constraint #1: Consistency of Enumeration, Description or combination for Conceptual and Value_Domains**

Suppose that r is an instance of the class *Value_Domain* and s is an instance of the class *Conceptual_Domain*, such that s is the meaning of r according to the *value_domain_meaning* association. There must exist such an s for every r according to the cardinality constraints on the *value_domain_meaning* association. Then it is either the case that r is an instance of *Enumerated_Value_Domain* and s is an instance of *Enumerated_Conceptual_Domain* or it is the case that r is an instance of *Described_Value_Domain* and s is an instance of *Described Conceptual_Domain*. Since neither *Value_Domains*, nor *Conceptual_Domains* are disjoint w.r.t. the Enumerated and Described subtypees it may be that r and s are both Enumerated and Conceptual Value/Conceptual_Domains.

**Constraint #2: Consistency of meanings reached by meaning associations**

Suppose that there exists an instance x of the class *Described_Value_Domain*, such that the instance y is the meaning of x according to the *value_domain_meaning* association (since every instance of a *Described_Value_Domain* is also a *Value_Domain*) where y is some instance of a *Conceptual_Domain* (either a *Described Conceptual_Domain* or an *Enumerated_Conceptual_Domain*). There must exist such an instance y according to the cardinality constraints on the value_domain_meaning association.

EDITOR'S NOTE #38. (Action required) The parenthetical comment above (either a *Described Conceptual_Domain* or an *Enumerated_Conceptual_Domain*) appears incorrect, since the following paragraph requires the Conceptual_Domain to be a Described_Conceptual_Domain. Can we remove the parenthetical comment?

According to the cardinality constraints for the described_value_meaning association there must also exist an instance z of the Described_Conceptual_Domain such that z is the meaning of x. Then it must be the case

that z is equal to y, i.e., the meaning of x must be same according to both the value_domain_meaning and described_value_domain meaning associations.

**Constraint #3: Mapping Enumerated_Value_Domains across Enumerated_Conceptual_Domains**

Suppose that there exists an instance *u* of the class *Enumerated_Value_Domain*, such that the instance *v* is the meaning of *u* according to the *value_domain_meaning* association (since every instance of a *Enumerated_Value_Domain* is also a *Value_Domain*) where v is some instance of a *Conceptual_Domain* (either a *Described Conceptual_Domain* or an *Enumerated_Conceptual_Domain*). There must exist such an instance *v* according to the cardinality constraints on the *value_domain_meaning* association.

EDITOR'S NOTE #39. (Action required) The parenthetical comment above (either a *Described Conceptual_Domain* or an *Enumerated_Conceptual_Domain*) appears incorrect, since the following paragraph requires the Conceptual_Domain to be an Enumerated_Conceptual_Domain. Can we remove the parenthetical comment?

Now for each instance u of the class *Enumerated_Value_Domain* there must exist a non-null set W of the members of the *Permissible_Values* class according to the *permissible_value_set* association. For each element $w_i$ of W there is an exactly one instance $m_i$ of the class *Value_Meaning* such that $m_i$ is the meaning of $w_i$ according to the *permissible_value_meaning* association. Let M be the set union of these $m_i$. Now consider the (possibly empty) sets $E_i$ each of which is the unions of instances of the class *Enumerated_Conceptual_Domain* which are the containing domains of the various value meanings of each $m_i$. Then it must be the case that for every $m_i$ in M there exists an instance e in the set $E_i$ such that e is equal to v.

NOTE The final existential quantification (rather than universal quantification) over the elements of each set $E_i$ arises because we no longer constrain *Value_Meanings* to exist in a single *Enumerated_Conceptual_Domain*.

#### 10.3.4.3 Consistent Dimensionalities

*Conceptual_Domains* may have an attribute *conceptual_domain_dimensionality*. *Value_Domains* may have an attribute *value_domain_unit_of_measure* of type *Unit_of_Measure*. Suppose that we have an instance c of the class *Conceptual_Domain* and an instance v of a *Value_Domain* such that c is the meaning of v according to the *value_domain_meaning* association (or some equivalent path as above). Suppose that d (of type *Dimensionality*) is the *conceptual_domain_dimensionaility* attribute of the instance c. Suppose that e is the dimensionality of the *value_domain_unit_of_measure* of v. Then it must be the case that the d is equal to e.

In plain English, the *dimensionality* of the *unit_of_measure* of a *Value_Domain* must be the same as the *dimensionality* of the *Conceptual_Domain* which provides the meaning of the *Value_Domain*.

## 10.4 Measurement region

### 10.4.1 Overview

The measurement region illustrated in Figure 10-4.

EDITOR'S NOTE #40. (Action required) Issue 124 proposes introducing a *Measurement_Class* to group units of measure.

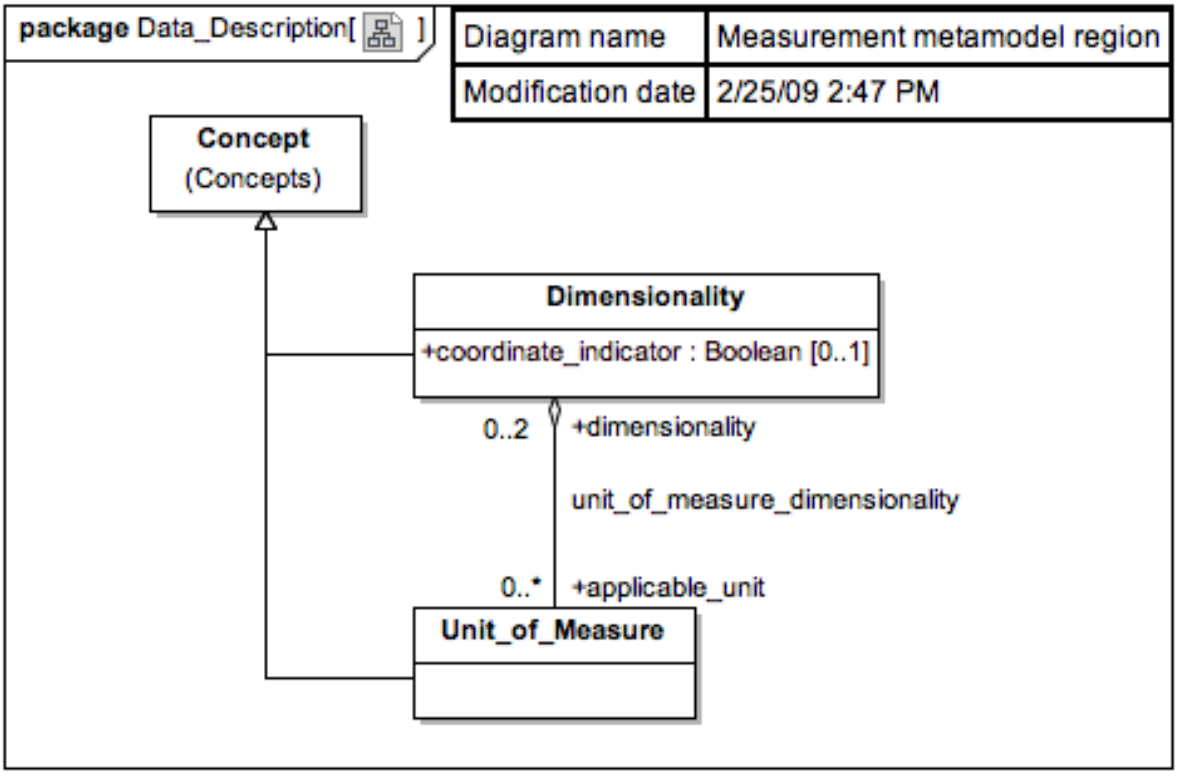



**Figure 10-4 — Measurement metamodel region**

### 10.4.2 Classes in the Measurement region

#### 10.4.2.1 Unit_of_Measure

EDITOR'S NOTE #41. (Action required) The resolution of Issue 125 states to add "unit_of_measure_scheme_reference" to "Unit_of_Measure" class, to identify source of "unit_of_measure_name". However, "unit_of_measure_name" was removed in favour of using the common facilities of Designatable_Item. Should we still add a scheme reference?

If appropriate, a *Value_Domain* (10.3.2.5) may be associated with a *Unit_of_Measure* — the units in which any associated *Data_Element* (10.5.2.1) values are specified. *Unit_of_Measure* may be named and defined by making it a *Designatable_Item* (6.2.2.2).

#### 10.4.2.2 Dimensionality

##### 10.4.2.2.1 Description of Dimensionality

*Dimensionality* is the class used to represent a set of equivalent units of measure, where equivalence between two units of measure is determined by the existence of a quantity-preserving one-to-one correspondence between values measured in one unit of measure and values measured in the other unit of measure, independent of context, and where the characterizing operations are the same.

A very common example is the use of temperature to measure the absolute temperature of a point, or to measure the size of a temperature interval, e.g., the temperature difference across the wall of a furnace. Aside from the semantic difference, the function for converting units of measure, e.g., temperature, depends on whether it is a coordinate or an interval measure. For example when converting degrees Celsius to Kelvins, one must add 273.16 for temperature coordinates, but not for temperature interval measures.

Note however, that in the Dimensionality class we do not explicitly specify what the frame of reference is for the Dimensionality. For some units of measure, such as temperature in Kelvins, or degrees Celsius the frame of reference is implicit in the units of measure. Additional examples of coordinate Dimensionalities would include longitude and latitude. However, in many cases the frame of reference for a coordinate measurement is specified as part of the Data_Element. This is quite common in computer aided design applications.

**EXAMPLE 1**: inches, feet, meters, and centimeters are all units of measure whose dimensionality is length. Other common dimensionalities include: mass, time, area, volume, etc.

NOTE 1 The equivalence defined here forms an equivalence relation on the set of all units of measure. Each equivalence class corresponds to a dimensionality. The units of measure "temperature in degrees Fahrenheit" and "temperature in degrees Celsius" have the same dimensionality, because given a value measured in degrees Fahrenheit there is a value measured in degrees Celsius that is the same quantity, and vice-versa. Quantity preserving one-to-one correspondences are the well-known equations $C^o = (5/9)*(F^o - 32)$ and $F^o = (9/5)*(C^o) + 32$. (Note that we have here assumed we are dealing with temperature coordinates. There is no offset when converting among temperature interval measures, e.g., the temperature difference between the coldest and hottest temperature on a day.)

NOTE 2 Units of measure are not limited to physical categories. Examples of physical categories are: linear measure, area, volume, mass, velocity, time duration. Examples of non-physical categories are: currency, quality indicator, color intensity.

NOTE 3 Quantities may be grouped together into categories of quantities which are mutually comparable. Lengths, diameters, distances, heights, wavelengths and so on would constitute such a category. Mutually comparable quantities usually have the same dimensionality (but see note 4) ISO 31-0 calls these "quantities of the same kind".

NOTE 4 The requirement of common “characterizing operations” for all units of measure with the same dimensionality is a stronger requirement than that commonly adopted in conventional dimensional analysis (where comparability and transformability usually suffice). Thus with respect to temperature, absolute temperature coordinates (e.g., Kelvins) are here considered to be a different dimensionality than “offset” temperature coordinates (e.g., degrees Celsius or Fahrenheit). It is meaningful to take the ratio of absolute temperature coordinates, but not of “offset” temperature coordinates, wherein the arbitrary translation of zero renders ratios meaningless. The notion of characterizing operations used here has been adapted from the statistics literature where distinctions are commonly made among categorical, ordered, interval, and ratio measures.

NOTE 5 Dimensionalities for physical units of measurement are commonly specified as the products or quotients of powers of basis dimensions: mass, length, time… However, in this metamodel we do not dictate the specification of dimensionalities, only their names and coordinate status.

#### 10.4.2.2.2 Attributes of Dimensionality

##### 10.4.2.2.2.1 coordinate_indicator

| | |
|---|---|
| Attribute name: | ***coordinate_indicator*** |
| Definition: | predicate on a *Dimensionality* whose value is true if the *Dimensionality* is a *coordinate.* |
| Obligation: | Conditional |
| Multiplicity: | *0..1* |
| Data type: | Boolean (5.1.2) |
| Condition: | The indicator must be specified for dimensionalities of physical units. |
| Note: | Otherwise, e.g., if the Dimensionality refers to an interval measure, the value of the coordinate element is false. |
| Example: | There might be two Dimensionalities concerned with length: one a measure of the size of an object (hence an interval measure), the other a measure of the location of an object (hence a coordinate). |

—— End of attributes of *Dimensionality* ——

### 10.4.3 Associations in the Measurement region

#### 10.4.3.1 unit_of_measure_dimensionality

*unit_of_measure_dimensionality* associates zero or more *Units_of_Measure* with up to two dimensionalities, one with a *coordinate_indicator* of *true*, the other with a *coordinate_indicator* of *false*.

*unit_of_measure_dimensionality* has two roles:

— *dimensionality* (verb form: has_dimensionality), which references an instance of the *Dimensionality* class;

— *applicable_unit* (verb form: has_applicable_unit), which references an instance of the *Unit_of_Measure* class.

Note: While units of measure are commonly physical units of measure, they may also be currency units (in which the corresponding Dimensionality would be money).

## 10.5 Data_Element region

### 10.5.1 Overview

The Data_Element metamodel region, illustrated in Figure 10-5 — Data_Element metamodel region, is used to address the administration of *Data_Elements*. *Data_Elements* provide the formal representations for some information (such as a fact, a proposition, an observation, etc.) about some concrete or abstract thing. *Data_Elements* are reusable and shareable representations of *Data_Element_Concepts*.

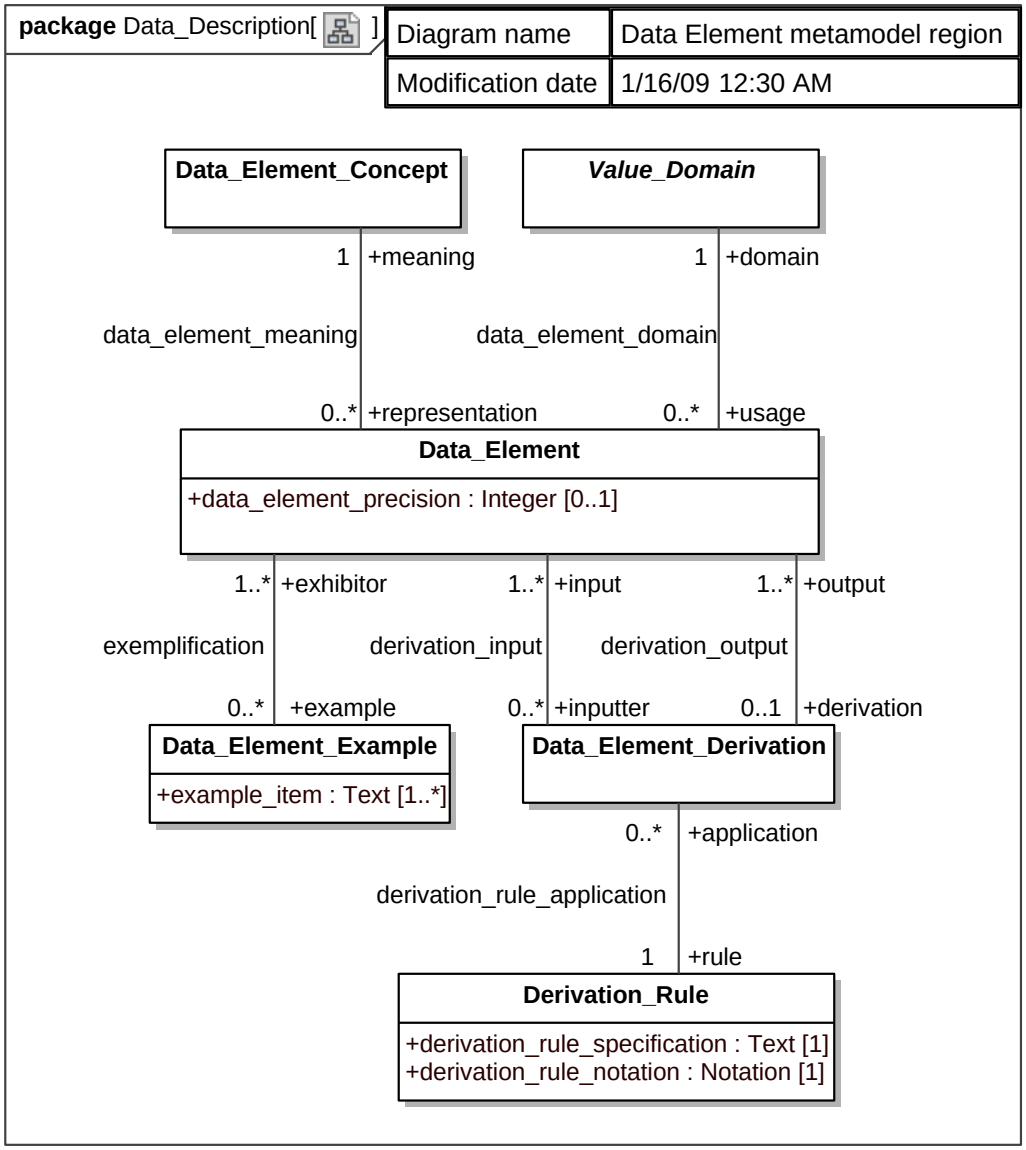



**Figure 10-5 — Data_Element metamodel region**

### 10.5.2 Classes in the Data_Element Region

#### 10.5.2.1 Data_Element

##### 10.5.2.1.1 Description of Data_Element

A *Data_Element* is considered to be a basic unit of data of interest to an organization. It is a unit of data for which the definition, identification, representation, and permissible values are specified by means of a set of attributes.

A *Data_Element* is formed when a *Data_Element_Concept* (10.5.2.2) is assigned a representation. One of the key components of a representation is the *Value_Domain* (10.5.2.3), i.e., restricted valid values.

A *Data_Element* is the association of a *Data_Element_Concept* with a *Value_Domain*. A *Data_Element* cannot be recorded in a *Metadata Registry* without being associated with both a *Data_Element_Concept* and a *Value_Domain*.

#### 10.5.2.1.2 Attributes of Data_Element

##### 10.5.2.1.2.1 *data_element_precision*

| | |
|---|---|
| Attribute name: | ***data_element_precision*** |
| Definition: | number of decimal places permitted in any associated data element values |
| Obligation: | Optional |
| Multiplicity: | *0..1* |
| Data type: | Integer (5.1.6) |

—— End of attributes of *Data_Element* ——

### 10.5.2.2 Data_Element_Concept

*Data_Element_Concept* is described under the Data_Element_Concept region in 10.2.2.3. A *Data_Element_Concept* may be associated with several *Value_Domains* resulting in a different *Data_Element* for each association.

### 10.5.2.3 Value_Domain

*Value_Domain* is described under the Conceptual_Domain and Value_Domain region in 10.3.2.5. A *Value_Domain* provides representation, but has no implication as to what *Data_Element_Concept* the values are associated with, nor what the values mean. A *Value_Domain* may be associated with multiple *Data_Elements*.

### 10.5.2.4 Data_Element_Example

#### 10.5.2.4.1 Description of Data_Element_Example

Every *Data_Element_Example* shall have an *exemplification* association with one or more *exhibitor Data_Elements*, where the *Data_Element_Example* serves as an *example* for the *Data_Element*.

A *Data_Element_Example* shall have one or more *example_item* attributes of type *Text* that provide representative illustrations of instances of a *Data_Element*.

#### 10.5.2.4.2 Attributes of Data_Element_Example

##### 10.5.2.4.2.1 *example_item*

| | |
|---|---|
| Attribute name: | ***example_item*** |
| Definition: | actual illustrative case of the *Data_Element* |
| Obligation: | Mandatory |
| Multiplicity: | *1..** |
| Data type: | Text (5.1.17) |

—— End of attributes of *Data_Element Example* ——

#### 10.5.2.5 Derivation_Rule

##### 10.5.2.5.1 Description of Derivation_Rule

A *Derivation_Rule* specifies the logical, mathematical, and/or other operations for derivation. The *Derivation_Rule* may range from a simple operation such as subtraction to a very complex set of derivations (derivation being defined as a relationship between a *Derivation_Rule* and an input set upon which it acts). *Derivation_Rules* are not limited to arithmetic and logical operations.

As a *Registered_Item,* a *Derivation_Rule* is directly or indirectly associated with an *Administration_Record* and can be identified, named, defined and optionally classified as a *Classifiable_Item* in a *Classification_Scheme*. A *Derivation_Rule* may be registered as an *Administered_Item* without necessarily being associated with any *Data_Element_Derivation*.

A *Derivation_Rule* may have a *derivation_rule_application* association with zero or more *application Data_Element_Derivations*, where the *Derivation_Rule* provides the *rule* for the associated *Data_Element_Derivation*.

Every *Derivation_Rule* must have exactly one *derivation_rule_specification* of type *Text* that specifies the rule semantics.

Every *Derivation_Rule* must have exactly one *derivation_rule_notation* of type *Notation* that specifies the syntax and semantics used in the *derivation_rule_specification.*

A *Derivation_Rule* may be registered as a *Registered_Item* without necessarily being associated with any *Data_Element_Derivation*.

##### 10.5.2.5.2 Attributes of Derivation_Rule

###### 10.5.2.5.2.1 *derivation_rule_specification*

| | |
|---|---|
| Attribute name: | ***derivation_rule_specification*** |
| Definition: | text of a specification of a *data element Derivation_Rule* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | Text (5.1.17) |

###### 10.5.2.5.2.2 *derivation_rule_notation*

| | |
|---|---|
| Attribute name: | ***derivation_rule_notation*** |
| Definition: | *notation* used to describe the *Derivation_Rule* |
| Obligation: | Mandatory |
| Multiplicity: | *1* |
| Data type: | Notation (5.1.9) |

—— End of attributes of *Derivation_Rule* ——

#### 10.5.2.6 Data_Element_Derivation

A *Data_Element_Derivation* is the application of a *Derivation_Rule* to one or more input *Data_Elements*, to derive one or more output *Data_Elements.* A *Data_Element_Derivation* may have a *Derivation_Rule* that is a specification of derivation for the *Data_Element*.

*Data_Element_Derivation* is an object that describes the *Data_Element(s)* that serve as sources or inputs to a *Derivation_Rule* and the *Data_Element(s)* that are the products or outputs of the *Derivation_Rule*.

Every *Data_Element_Derivation* shall have one or more *derivation_input* associations with an *input Data_Element*, where the *Data_Element_Derivation* serves as the *inputter* for the associated *Data_Element*.

EDITOR'S NOTE #42. (Action required) Can we find a better name for the role *inputter?*

Every *Data_Element_Derivation* shall have a one or more *derivation_output* associations with an *output Data_Element*, where the *Data_Element_Derivation* serves as the *derivation* for the associated *Data_Element*.

### 10.5.3 Associations in the Data_Element region

#### 10.5.3.1 data_element_domain Association

EDITOR'S NOTE #43. (Action required) issue 230 recommends renaming this association to 'data element value domain' to more clearly distinguish it from 'data element concept domain'.

*data_element_domain* is an association between a *Data_Element* and a *Value_Domain* that describes a set of possible values that may be recorded in an instance of the *Data_Element*.

#### 10.5.3.2 data_element_meaning Association

*data_element_meaning* is an association between a *Data_Element* and a *Data_Element_Concept* that identifies the *Data_Element_Concept* that provides the meaning for the *Data_Element*.

#### 10.5.3.3 exemplification Association

*Exemplification* is an association between a *Data_Element* and a *Data_Element_Example* that provides an example instance or use of the *exhibitor Data_Element*. *Exemplification* has two roles: *exhibitor* and *example*. The *exhibitor* role refers to a *Data_Element* and the *example* role refers to a *Data_Element_Example*. An *exhibitor Data_Element* may be associated with zero or more *example Data_Element_Examples*. Every *Data_Element_Example* shall be associated with one or more *exhibitor Data_Elements*.

#### 10.5.3.4 derivation_input Association

*derivation_input* is an association between a *Data_Element* and a *Data_Element_Derivation*. That indicates that the *input Data_Element* is a source for the *Data_Element_Derivation*. *Derivation_input* has two roles: *input* and *inputter*. The *input* role refers to a *Data_Element* and the *inputter* role refers to a *Data_Element_Derivation*. An *input Data_Element* may be associated with zero or more *inputter Data_Element_Derivations*. Every *Data_Element_Derivation* shall be associated with one or more *input Data_Elements*.

#### 10.5.3.5 derivation_output Association

*derivation_output* is an association between a *Data_Element* and a *Data_Element_Derivation* that indicates that the *output Data_Element* is the result of the application of a *Data_Element_Derivation*. *Derivation input* has two roles: *output* and *derivation*. The *output* role refers to a *Data_Element* and the *derivation* role refers to a *Data_Element_Derivation*. An *output Data_Element* may be associated with zero or more *derivation Data_Element_Derivations*. Every *Data_Element_Derivation* shall be associated with one or more *output Data_Elements*.

#### 10.5.3.6 derivation_rule_application Association

*derivation_rule_application* is an association between a *Data_Element_Derivation* and a *Derivation_Rule* that specifies the *Derivation_Rule* that is utilized for the *Data_Element_Derivation*. *Derivation rule application* has two roles: *application* and *rule*. The *application* role refers to a *Data_Element_Derivation* and the *rule* role refers to a *Derivation_Rule*. Every *application Data_Element_Derivation* must be associated with exactly one *rule Derivation_Rule.* A *Derivation_Rule* may be associated with zero or more *application Data_Element_Derivations*.

## 10.6 Consolidated Data Description Metamodel

A consolidated metamodel is shown in Figure 10-6 — Consolidated Data Description metamodel. This combines the Data_Element_Concept, Data_Element, and Conceptual and Value_Domain regions of the model.

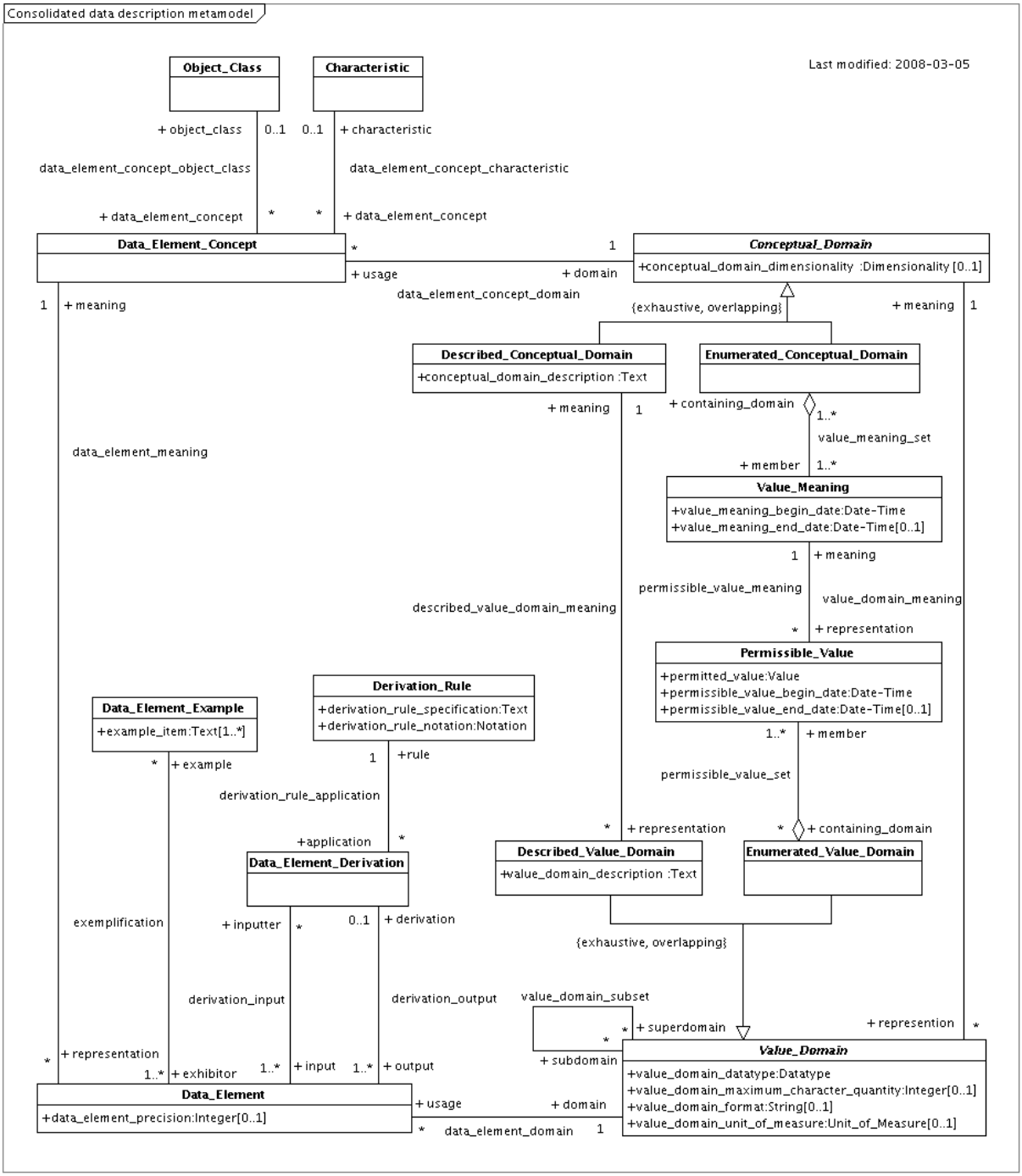



**Figure 10-6 — Consolidated Data Description metamodel**

## 10.7 Types of Concepts in the Data Description Metamodel

Figure 10-7 — Types of Concepts shows the classes in the Data Description package which are types of *Concepts* (i.e. they are sub-typed from the *Concept* class).

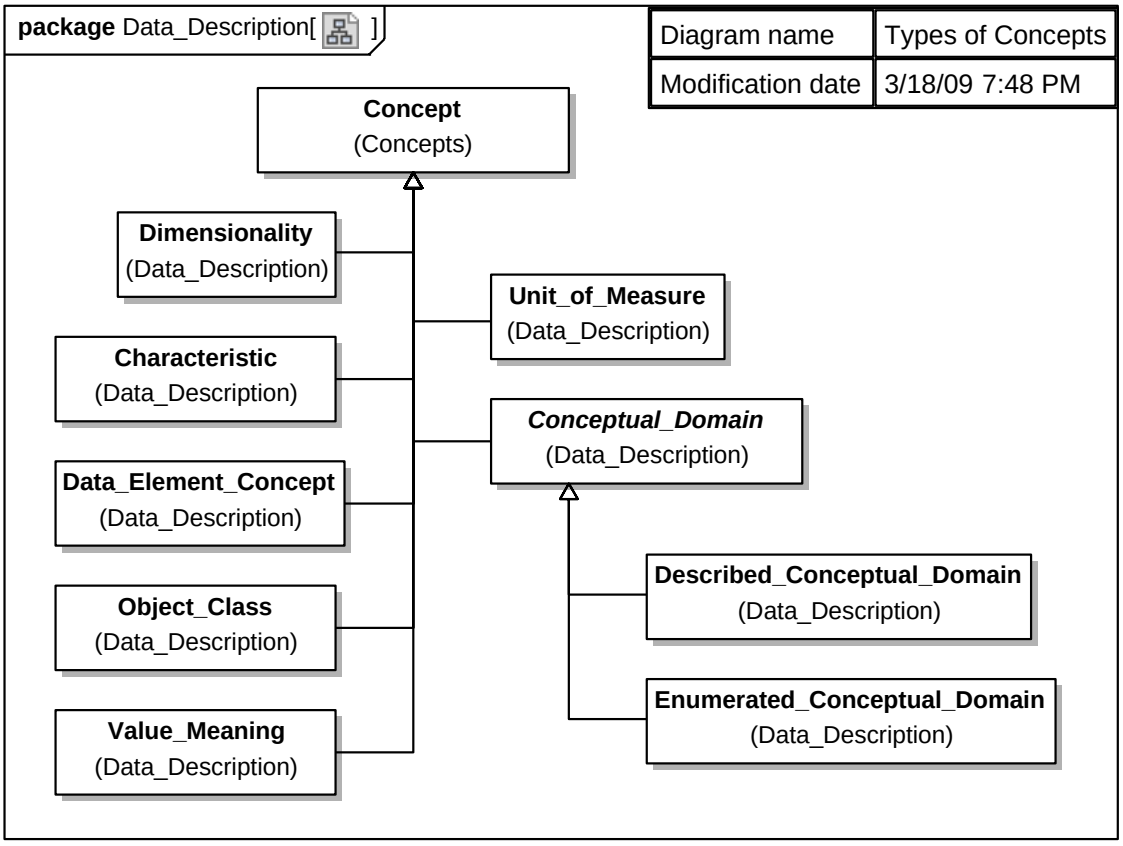



**Figure 10-7 — Types of Concepts in the Data Description package**

# 11 Basic attributes

EDITOR'S NOTE #44. (Action required) This clause has been carried over from Edition 2 without significant modification. Changes are required to reflect the changes in the model. Revisions will be made for the FCD, once the model has stabilized.

## 11.1 Use of basic attributes

This Clause is intended to provide continuity from ISO/IEC 11179-3:1994, which edition focused on basic attributes of data elements. However, the scope of this Clause extends beyond just data elements, to include: data element concepts, conceptual domains, value domains, permissible values and value meanings.

EDITOR'S NOTE #45. (Action required) The following sentence will need to be revised to reflect the changes made to the mappings among editions of the standard.

A mapping among the 1994 basic attributes, the 2002 basic attributes and the 2002 metamodel can be found in Annex D.

Clauses 5 through 10 describe a model for specifying metadata in a registry. However, sometimes the requirement for metadata specification exists outside the context of a registry, for example as part of an International Standard.

A specification of metadata consists of a set of attributes, and relationships among those attributes. This Clause specifies a set of *basic* attributes to be used in contexts other than a metadata registry. *Basic* means that they are frequently needed to specify a metadata item. The attributes specified in this Clause are also considered *basic* in the sense that additional attributes may be required when the metadata items are used in a particular context.

*Basic* does not imply that all standardized attributes presented in this Clause are required in all cases. Distinction is made between those basic attributes that are:

— mandatory: always required;

— conditional: required to be present under certain specified conditions;

— optional: permitted but not required.

NOTE The obligations specified for some basic attributes (especially identifiers) in contexts other than a registry are different from those specified for metadata items in a registry, as defined in Clauses 4 through 10.

## 11.2 Common attributes

The attributes listed in this subclause are common to all types of Administered_Item. These attributes are further categorized as: Identifying, Definitional, Administrative, and Relational.

### 11.2.1 Identifying

EDITOR'S NOTE #46. (Action required) We should probably distinguish Identification from Designation, as we do in Clause 4.

| Attribute | Obligation |
|---|---|
| *name* | One or more per metadata item *(see note 1).* |
| *context name* | Zero or more per metadata item. Required if more than one *name* attribute exists. |
| *context identifier* | Zero or one per metadata item. Required if *context name* is not unique within its usage context (e.g. a standard). |
| *context description* | One per *context name*. |
| *item identifier* | Zero or one per metadata item. Required if *name* is not unique within a given *context (see note 2).* |
| *item identifier – data identifier* | One per *item identifier*. (The mandatory portion of an *item identifier*.) |
| *item identifier – item registration authority identifier* | Zero or one per *item identifier*. (The optional portion of an *item identifier - see note 3.)* |
| *version* | Zero or one per metadata item *(see note 4).* |

NOTE 1 If more than one *name* is specified within a given *context*, it is usual nominate one name as "preferred", and the others as "synonyms".

NOTE 2 While *item identifier* is mandatory within a registry (see 4.8.1.4), it is only conditional in non-registry usages. The requirement for an *item identifier* can be eliminated by qualifying *name* and/or *context name* to ensure that the combination is unique.

NOTE 3 While *item registration authority identifier* is mandatory within a registry (see 4.8.1.4), it is optional in nonregistry settings.

NOTE 4 Within a registry, *version* is part of an *item identifier*. In non-registry settings, *version* may be used independently of *item identifier*.

### 11.2.2 Definitional

| Attribute | Obligation |
|---|---|
| *definition* | One for each *context* in which the metadata item is used *(see note 1).* |
| *definition_language_identifier* | Zero or one per *definition.* |
| *definition_source_reference* | Zero or one per *definition.* |

NOTE Where multiple *definitions* are assigned to the same metadata item, the semantics of the *definition* should be the same across all *contexts*. (If the semantics are different, separate metadata items should be specified.) However, the terminology used to express the semantics may need to be different in different *contexts*, and thus separate *definitions* are permitted for each *context*.

### 11.2.3 Administrative

Administrative attributes are primarily associated with recording metadata items in a registry. They are therefore optional in non-registry settings.

| Attribute | Obligation |
|---|---|
| *comments* | Zero or one per metadata item. |
| *registration_status* | Zero or one per metadata item. |
| *responsible organization name* | Zero or one per metadata item. |
| *submitting organization name* | Zero or one per metadata item. |

### 11.2.4 Relational

| Attribute | Obligation |
|---|---|
| *classification scheme name* | One for each *classification scheme* in which a metadata item is classified. |
| *classification scheme identifier* | Zero or one per *classification scheme name*. Required if *classification scheme name* is not unique within a *context*. |
| *classification scheme type name* | One for each *classification scheme* in which a metadata item is classified. |
| *classification scheme item type name* | Zero or one for each *classification scheme* in which a metadata item is classified (see note 1). |
| *classification scheme item value* | One for each *classification scheme item* by which a metadata item is classified. |
| *related metadata reference* | Zero or more per metadata item (see note 2). |
| *type of relationship* | One per *related metadata reference*. |

NOTE 1 The metamodel in 0 treats *keywords* as a type of *classification scheme*.

NOTE 2 A *Registration_Authority* could choose to use a *Reference_Document*, an *administrative_note* or an *explanatory_comment* to record a *related metadata reference*.

## 11.3 Attributes specific to Data_Element_Concepts

The attributes listed in this subclause are specific to Data_Element_Concepts.

| Attribute | Obligation |
|---|---|
| *object class name* | One per *data element concept.* |
| *object class identifier* | Zero or one per *object class name.* |
| *property name* | One per *data element concept.* |
| *property identifier* | Zero or one per *property name.* |

### 11.4 Attributes specific to Data_Elements

The attributes listed in this subclause are specific to Data_Elements.

EDITOR'S NOTE #47. (Action required) Issue 114 has removed *Representation_Class.* We need some text to explain the use of a *Classification_Scheme* instead.

| **Attribute** | **Obligation** |
|---|---|
| *Value domain name* | Zero or one per *data element.* |
| *Value domain identifier* | Zero or one per *data element.* |
| *Datatype name* | Zero or one per *data element.* Required if neither *value domain name* nor *value domain identifier* is not specified. |
| *Datatype scheme reference* | Zero or one per *datatype_name.* |
| *Layout of representation* | Zero or one per *data element.* |
| ~~*Representation class*~~ | ~~Zero or one per *data element.*~~ |
| *Maximum size* | Zero or one per *data element.* |
| *Minimum size* | Zero or one per *data element.* |

### 11.5 Attributes specific to Conceptual_Domains

The attributes listed in this subclause are specific to Conceptual_Domains.

| **Attribute** | **Obligation** |
|---|---|
| *dimensionality* | Zero or one per *conceptual domain.* |

### 11.6 Attributes specific to Value_Domains

The attributes listed in this subclause are specific to Value_Domains.

| **Attribute** | **Obligation** |
|---|---|
| *datatype_name* | One per *value domain.* |
| *datatype_scheme_reference* | Zero or one per *datatype_name.* |
| *unit of measure name* | Zero or one per *value domain.* |

### 11.7 Attributes specific to Permissible_Values

The attributes listed in this subclause are specific to Permissible_Values.

| Attribute | Obligation |
|---|---|
| *value* | One per *permissible value.* |
| *permissible_value_begin_date* | Zero or one per *permissible value.* |
| *permissible_value_end_date* | Zero or one per *permissible value.* |

### 11.8 Attributes specific to Value_Meanings

The attributes listed in this subclause are specific to Value_Meanings.

| Attribute | Obligation |
|---|---|
| *value meaning description* | One per *value meaning.* |
| *value meaning identifier* | Zero or one per *value meaning.* |
| *value_meaning_begin_date* | Zero or one per *value meaning.* |
| *value_meaning_end_date* | Zero or one per *value meaning.* |

# 12 Conformance

## 12.1 Overview of Conformance

This part of ISO/IEC 11179 prescribes a conceptual model, not a physical implementation. Therefore, the metamodel need not be physically implemented exactly as specified. However, it must be possible to unambiguously map between the implementation and the metamodel in both directions.

This part of ISO/IEC 11179 also prescribes a list of basic attributes (see clause 11) for situations where a full conceptual model is not required or not appropriate.

Conformance may be claimed to either the conceptual model, or the basic attributes or both. Conformance claims shall specify a Degree and a Level of Conformance, as described below.

Conformance statements with respect to this standard must also be explicit as to which portions of this standard conformity is being claimed. This may done in some cases simply by reference to one or more of the clauses (see 12.3 Conformance by Clause). In other cases, conformance may instead be claimed to one or more of the standard profiles (see 12.4 Registry Conformance), which specify combinations of multiple clauses, and how they are to be fitted together.

When a *registry product* makes a conformance claim, the product must support all the associated functionality, and must enable the enforcement of the associated constraints. When a *registry instance* makes a conformance claim, it must actually enforce those constraints.

## 12.2 Degree of Conformance

The distinction between “strictly conforming” and “conforming” implementations is necessary to address the simultaneous needs for interoperability and extensions. This part of ISO/IEC 11179 describes specifications that promote interoperability. Extensions are motivated by needs of users, vendors, institutions, and industries, and:

a) are not directly specified by this part of ISO/IEC 11179,

b) are specified and agreed to outside this part of ISO/IEC 11179, and

c) may serve as trial usage for future editions of this part of ISO/IEC 11179.

A strictly conforming implementation may be limited in usefulness but is maximally interoperable with respect to this part of ISO/IEC 11179. A conforming implementation may be more useful, but may be less interoperable with respect to this part of ISO/IEC 11179.

### 12.2.1 Strictly conforming implementations

A strictly conforming implementation:

a) shall support all mandatory, optional and conditional data element attributes and relationships;

b) shall not use, test, access, or probe for any extension features nor extensions to data element attributes;

c) shall not recognize, nor act on, nor allow the production of data element attributes that are dependent on any unspecified, undefined, or implementation-defined behavior.

NOTE The use of extensions to the metamodel or the basic attributes may cause undefined behavior.

#### 12.2.2 Conforming implementations

A conforming implementation:

a) shall support all mandatory, optional and conditional data element attributes and relationships;

b) as permitted by the implementation, may use, test, access, or probe for extension features or extensions to data element attributes;

c) may recognize, act on, or allow the production of data element attributes that are dependent on implementation-defined behavior.

NOTE 1 All strictly conforming implementations are also conforming implementations.

NOTE 2 The use of extensions to the metamodel or the basic attributes may cause undefined behavior.

### 12.3 Conformance by Clause

Conformance claims may be limited to individual clauses 6-11 of this standard. Clauses 6-10 are all dependent upon one or more other clauses of this standard (see Figure 4-1 — Package dependencies), so conformance to any of these clauses must be understood to imply conformance also to relevant provisions specified in one or more preceding clauses.

For clauses 6 and 8-10, conformance may be claimed for a set of data structures and/or datatypes.

For clauses 7 and 11, conformance may be claimed for a register or registry (a set of administered metadata), or for a registry software system.

### 12.4 Registry Conformance

#### 12.4.1 Overview

Registries shall claim conformance to either clause 7 or clause 11. Registries conforming to clause 7 may be more specifically claimed to conform to one of the standard profiles specified in clause 12.4.2, each of which additionally incorporates some subset of the clauses 6 and 8-10.

The table below compares the range of registry conformance for Edition 3, to the conformance requirements found in prior Editions of the standard.

| Conformance Level | Edition 3 | Edition 2 Level 2 | Edition 2 Level 1 | Edition 1 |
|---|---|---|---|---|
| Basic package | Mandatory | Mandatory | N/a | N/a |
| Identification, Designation and Definition package | Mandatory | Mandatory | N/a | N/a |
| Registration package | Mandatory | Mandatory | N/a | N/a |
| Concepts package | Mandatory | Mandatory | N/a | N/a |
| Binary_Relations package | Optional | N/a | N/a | N/a |
| Data Description package | Optional | Mandatory | N/a | N/a |
| Basic attributes | Optional | N/a | Mandatory | Mandatory |

**Table 12-1 – Comparison for Conformance Levels across Editions of ISO/IEC 11179-3**

#### 12.4.2 Standard Profiles for Edition 3 Registries

Implementors of either a generic registry platform (software system) which is customizable for a range of registered content types, or of a registry or registry software system supporting some specific range of

registered content types not specified in this standard, may simply claim conformance to clause 7. Alternatively, the following standard profiles are defined, specifying how additional clauses should be combined with clause 7:

— **Concept Systems Registry**: Implements clauses 6-8, and also satisfies the following additional provisions:

1. All Concepts and Concept_Systems must be Registered_Items.
2. All Registered_Items must also be Designatable_Items and Classifiable_Items.

— **Extended Concept Systems Registry**: Implements clause 9, in addition to all provisions of the Concept Systems Registry profile.

— **Metadata Registry**: Implements clause 10, in addition to all provisions of the Concept Systems Registry profile, and also satisfies the following additional provision:

1. All Data_Elements, Value_Domains, and Derivation_Rules must be Registered_Items.

— **Extended Metadata Registry**: Implements all of clauses 6-10, and also satisfies all provisions of the Metadata Registry profile.

#### 12.4.3 Conformance Levels for Edition 2 Level 2

The closest profile to Edition 2 Level 2 is the **Metadata Registry** profile above.

#### 12.4.4 Conformance Levels for Edition 2 Level 1 and Edition 1

Only those metadata elements, relationships and properties specified in Clause 11 are supported and used.

### 12.5 Obligation

Properties and relationships specified in this part of ISO/IEC 11179 are stated to be Mandatory, Conditional or Optional.

For the purpose of conformance:

a) Mandatory properties and relationships shall exist, and shall conform to the provisions of this part of ISO/IEC 11179.

b) Anything specified as Conditional within this part of ISO/IEC 11179 shall be treated as Mandatory if the associated condition is satisfied, and shall otherwise be not present.

c) Optional properties and relationships are not required to exist, but if they do exist they shall conform to the provisions of this part of ISO/IEC 11179.

Such obligation is enforced if and only if the Registration Status of the associated metadata items is Recorded or higher.

### 12.6 Implementation Conformance Statement (ICS)

An implementation claiming conformance to this part of ISO/IEC 11179 shall include an Implementation Conformance Statement stating:

a) whether it conforms or strictly conforms (12.2 Degree of Conformance);

b) which clauses are supported (12.3 Conformance by Clause);

c) which registry type is supported (12.4 Registry Conformance)

d) what extensions, if any, are supported or used.

## 12.7 Roles and Responsibilities for Registration

Conformance needs to be considered in the context of the roles and responsibilities of registration authorities, as covered by ISO/IEC 11179-6: Registration of data elements.

Extended conformance of systems requires formalisation of procedures, agreement of roles and responsibilities between parties, and guidelines addressing use of software products and conversions from other systems. The formalisation of these aspects must be consistent with the conformance requirements in the above Clauses, and roles of registration authorities as set out in ISO/IEC 11179-6.

# Annex A
(normative)

# Alphabetical List of Terms

| Term | Defined in |
|---|---|
| Acceptability (datatype) | 6.2.2.1 |
| acceptability rating | 3.3.1 |
| administered item | 3.4.1 |
| Administered_Item (class) | 7.1.2.2 |
| administrative_note ( attribute of *Registration_Record*) | 7.1.2.6.2.4 |
| administrative_status ( attribute of *Registration_Record*) | 7.1.2.6.2.6 |
| antisymmetric relation | 3.4.2 |
| arity (attribute of *Relation)* | 8.1.2.4.2.1 |
| assertion | 3.4.3 |
| Assertion (class) | 8.1.2.3 |
| assertion_formula (attribute of Assertion) | 8.1.2.3.2.1 |
| assertion_inclusion (association) | 8.1.3.5 |
| assertion_term (association) | 8.1.3.6 |
| association | 3.2.1 |
| association class | 3.2.2 |
| asymmetric relation | 3.4.4 |
| attached item | 3.4.5 |
| Attached_Item (class) | 7.1.2.3 |
| attachment (association) | 7.1.6.1 |
| attribute | 3.2.3 |
| attribute instance | 3.3.2 |
| attribute value | 3.3.3 |
| authority rule (attribute of *Naming_Convention*) | 3.3.12, 6.2.2.7.2.2 |
| basic attribute | 3.3.4 |
| binary relation | 3.4.6 |
| Binary_Relation (class) | 9.1.2.2 |
| binding | 3.3.5 |
| boolean | 3.4.7 |
| Boolean (datatype) | 5.1.2 |
| CD | 3.5.1 |
| change description | 7.1.2.2.2.4 |
| characteristic | 3.4.8 |
| Characteristic (class) | 10.2.2.2 |

| Term | Defined in |
|---|---|
| class | 3.2.4 |
| classifiable item | 3.4.10 |
| Classifiable_Item (type) | 8.2.2.1 |
| Classification (association) | 8.2.3.1 |
| classification scheme | 3.4.10 |
| classification_scheme (association) | 8.2.4.1 |
| common attribute | 3.3.7 |
| composite attribute | 3.2.5 |
| composite datatype | 3.2.6 |
| concept | 3.4.11 |
| Concept (class) | 8.1.2.1, 8.2.2.3 |
| concept system | 3.4.12 |
| Concept_System (class) | 8.1.2.2, 8.2.2.2 |
| concept_system_importation (association) | 8.1.3.4 |
| concept_system_membership (association) | 8.1.3.1, 8.2.4.2 |
| concept_system_notation (attribute of *Concept_System)* | 8.1.2.2.2.1 |
| concept_system_reference (association) | 8.1.3.3 |
| concept_system_source (association) | 8.1.3.2 |
| conceptual data model | 3.3.8 |
| conceptual domain | 3.4.13 |
| Conceptual_Domain (class) | 10.1.2.2,10.3.2.1 |
| conceptual_domain_description (attribute of *Described_Conceptual_Domain)* | 10.3.2.4.2.1 |
| conceptual_domain_dimensionality (attribute of *Conceptual_Domain)* | 10.3.2.1.2.1 |
| conditional | 3.3.9 |
| contact | 3.4.14 |
| Contact (class) | 5.1.3, 7.1.3.1 |
| contact_email (attribute of Contact) | 5.1.3.2.6 |
| contact_individual (attribute of Contact) | 5.1.3.2.1 |
| contact_mail_address (attribute of Contact) | 5.1.3.2.4 |
| contact_organization (attribute of Contact) | 5.1.3.2.2 |

| Term | Defined in |
|---|---|
| contact_phone (attribute of Contact) | 5.1.3.2.5 |
| contact_title (attribute of Contact) | 5.1.3.2.3 |
| context | 3.4.17 |
| Context (class) | 6.2.2.5 |
| coordinate | 3.3.10 |
| coordinate_indicator (attribute of *Dimensionality)* | 10.4.2.2.2.1 |
| creation_date (attribute of (*Registration_Record*) | 7.1.2.2.2.2 |
| data | 3.3.11 |
| data element | 3.4.18 |
| Data_Element | 10.1.2.4,10.5.2.1 |
| data element concept | 3.4.19 |
| Data_Element_Concept | 10.1.2.5,10.2.2.3 |
| data element concept characteristic | 3.4.20 |
| data_element_concept_characteristic (association) | 10.2.3.1 |
| data element concept domain | 3.4.21 |
| data_element_concept_domain (association) | 10.2.3.2 |
| data element concept object class | 3.4.22 |
| data_element_concept_object_class (association) | 10.2.3.3 |
| data element derivation | 3.4.23 |
| Data_Element_Derivation (association class) | 10.5.2.6 |
| data element domain | 3.4.24 |
| data_element_domain (association) | 10.5.3.1 |
| data element example | 3.4.25 |
| Data_Element_Example (class) | 10.5.2.4 |
| data element meaning | 3.4.26 |
| data_element_meaning (association) | 10.5.3.2 |
| data element precision | 3.4.27 |
| data_element_precision (attribute of *Data_Element*) | 10.5.2.1.2.1 |
| data model | 3.3.12 |
| datatype | 3.2.7 |
| Datatype | 10.3.2.9 |
| datatype_annotation (attribute of *Datatype)* | 10.3.2.9.2.4 |
| datatype_description (attribute of *Datatype)* | 10.3.2.9.2.2 |
| datatype_name (attribute of *Datatype)* | 10.3.2.9.2.1 |
| datatype_scheme_reference (attribute of *Datatype)* | 10.3.2.9.2.3 |
| Date (datatype) | 5.1.4 |

| Term | Defined in |
|---|---|
| date and time | 3.4.28 |
| Datetime (datatype) | 5.1.4 |
| DE | 3.5.2 |
| DEC | 3.5.3 |
| definition | 3.3.13 |
| definition (designatable item) | 3.4.29 |
| Definition (class) | 6.2.2.4 |
| definition_acceptability | 6.2.3.1.2.1 |
| definition context | 3.4.30 |
| Definition_Context (association class) | 6.2.3.1 |
| definition language | 3.4.31 |
| definition_language (attribute of *Definition*) | 6.2.2.4.2.2 |
| definition source reference | 3.4.32 |
| definition_source_reference (attribute of *Definition*) | 6.2.2.4.2.3 |
| definition text | 3.4.33 |
| definition_text (attribute of *Definition*) | 6.2.2.4.2.1 |
| derivation_input (association) | 10.5.3.4 |
| derivation_output (association) | 10.5.3.5 |
| derivation rule | 3.4.34 |
| Derivation_Rule (class) | 10.5.2.5 |
| derivation_rule_application (association) | 10.5.3.6 |
| derivation rule notation | 3.4.35 |
| derivation_rule_notation (attribute of *Derivation_Rule)* | 10.5.2.5.2.2 |
| derivation rule specification | 3.4.36 |
| derivation_rule_specification (attribute of *Derivation_Rule)* | 10.5.2.5.2.1 |
| described conceptual domain | 3.4.37 |
| Described Conceptual_Domain (Class) | 10.3.2.4 |
| described value domain | 3.4.38 |
| Described_Value_Domain (class) | 10.3.2.8 |
| described_value_domain_meaning (association) | 10.3.3.3 |
| designatable item | 3.4.39 |
| Designatable_Item (type) | 6.2.2.2 |
| designation | 3.3.14 |
| designation <designatable item> | 3.4.40 |
| Designation (class) | 6.2.2.3 |
| designation acceptability | 3.4.41 |
| designation_acceptability (attribute of *Designation_Context)* | 6.2.3.2.2.1 |

| Term | Defined in |
|---|---|
| designation context | 3.4.42 |
| Designation_Context (class) | 6.2.3.2 |
| designation language | 3.4.43 |
| designation_language (attribute of *Designation)* | 6.2.2.3.2.2 |
| designation namespace | 3.4.44 |
| designation_namespace (association) | 6.2.4.1 |
| designation sign | 3.4.45 |
| designation_sign (attribute of *Designation*) | 6.2.2.3.2.1 |
| dimensionality | 3.4.46 |
| Dimensionality | 10.4.2.2 |
| documentation_language_identifier <Registration_Authority> | 7.1.2.5.2.2 |
| effective_date <Registration_Record> | 7.1.2.6.2.2 |
| entity | 3.3.15 |
| enumerated conceptual domain | 3.4.47 |
| Enumerated_Conceptual_Domain (class) | 10.3.2.2 |
| enumerated value domain | 3.4.48 |
| Enumerated_Value_Domain (class) | 10.3.2.6 |
| example_item (attribute of *Data_Element_Example*) | 10.5.2.4.2.1 |
| exemplification (association) | 10.5.3.3 |
| explanatory_comment (attribute of *Administered_Item*) | 7.1.2.2.2.5 |
| extension | 3.3.16 |
| extension_identifier (attribute of *Language_Identification*) | 5.1.7.2.5 |
| full_expansion <scoped identifier> | 6.1.2.2.2.2 |
| generalization | 3.2.8 |
| identification (association) | 6.1.3.1 |
| identified item | 3.4.49 |
| Identified_Item (type) | 6.1.2.1 |
| identifier <metamodel> | 3.2.9 |
| identifier <Scoped_Identifier> | 6.1.2.2.2.1 |
| identifier_scope | 6.1.3.2 |
| individual | 3.4.50 |
| Individual (class) | 5.1.5 |
| integer | 3.4.51 |
| Integer (datatype) | 5.1.6 |
| international code designator | 3.4.52 |
| international_code_designator (attribute of *Registration_Authority_Identifier)* | 5.1.14.2.1 |

| Term | Defined in |
|---|---|
| item_definition | 6.2.4.2 |
| item_designation | 6.2.4.3 |
| item_slot | 6.1.3.3 |
| language | 3.3.17 |
| Language_Identification (class) | 5.1.7 |
| language_identifier (attribute of *Language_Identification)* | 5.1.7.2.1 |
| last_change_date (attribute of *Administered_Item*) | 7.1.2.2.2.3 |
| lexical_rule (attribute of *naming_Convention*) | 6.2.2.7.2.5 |
| Link (class) | 8.1.2.6 |
| Link_End (class) | 8.1.4.1 |
| link_end_role (association) | 8.1.3.9 |
| mail_address (attribute of *Organization*) | 5.1.10.2.2 |
| management | 3.4.53 |
| mandatory | 3.3.18 |
| mandatory_naming_convention_indicator (attribute of *Name_Space*) | 6.1.2.3.2.4 |
| MDR | 3.5.4 |
| metadata | 3.3.19 |
| metadata item | 3.3.20 |
| metadata object | 3.3.21 |
| metadata register | 3.3.22 |
| Metadata Registry (MDR) | 3.3.23 |
| metadata set | 3.3.24 |
| metamodel | 3.3.25 |
| metamodel construct | 3.3.26 |
| multiplicity (Attribute of *Relation_Role*) | 8.1.2.5.2.1 |
| name | 3.3.27 |
| name (attribute of *Individual*) | 5.1.5.2.1 |
| name (attribute of *Organization*) | 5.1.10.2.1 |
| namespace | 3.4.54 |
| Namespace (class) | 6.1.2.3, 6.2.2.6, 7.1.4.1 |
| naming_authority (Attribute of *Namespace*) | 6.1.2.3.2.1 |
| naming convention | 3.4.55 |
| Naming_Convention (class) | 6.2.2.7 |
| naming_convention_conformance (association) | 6.2.4.4 |
| naming_convention_utilization (association) | 6.2.4.5 |

| Term | Defined in |
|---|---|
| Natural_Range (datatype) | 5.1.8 |
| notation | 3.4.56 |
| Notation (class) | 5.1.9 |
| object | 3.3.28 |
| object class | 3.4.57 |
| Object_Class (class) | 10.2.2.1 |
| one_item_per_name_indicator (attribute of *Namespace*) | 6.1.2.3.2.3 |
| one_name_per_item_indicator (attribute of *Namespace*) | 6.1.2.3.2.2 |
| opi | 3.4.60, 3.5.5 |
| opi_source | 5.1.14.2.4 |
| optional | 3.3.29 |
| ordinal | 8.1.2.5.2.2 |
| organization | 3.4.58 |
| Organization | 5.1.10, 7.1.3.2 |
| organization identifier | 3.4.59 |
| organization_identifier | 5.1.14.2.2 |
| organization part | 3.3.30 |
| organization part identifier (opi) | 3.4.60 |
| organization_part_identifier | 5.1.14.2.3 |
| origin ( attribute of *Administered_Item*) | 7.1.2.2.2.6 |
| ORM | 3.5.6 |
| OWL | 3.5.7 |
| OWL-DL | 3.5.8 |
| permissible value | 3.4.61 |
| Permissible_Value | 10.3.2.7 |
| permissible_value_begin_date (Attribute of *Permissible_Value*) | 10.3.2.7.2.2 |
| permissible_value_end_date (Attribute of *Permissible_Value*) | 10.3.2.7.2.3 |
| permissible_value_meaning (association) | 10.3.3.4 |
| permissible_value_set (association) | 10.3.3.5 |
| permitted_value (Attribute of *Permissible_Value*) | 10.3.2.7.2.1 |
| phone number | 3.4.62 |
| Phone_Number (class) | 5.1.11 |
| postal address | 3.4.63 |
| Postal_Address (class) | 5.1.12 |
| previous_state <Registration_Record> | 7.1.2.6.2.7 |
| primitive datatype | 3.2.10 |
| private_use_qualifier | 5.1.7.2.6 |
| quantity | 3.3.31 |

| Term | Defined in |
|---|---|
| RA | 3.5.9 |
| ra_identifier | 7.1.2.5.2.1 |
| RDF | 3.5.10 |
| Reference (association class) | 7.1.5.2 |
| reference document | 3.4.64 |
| Reference_Document (class) | 5.1.13, 7.1.3.3 |
| reference document identifier | 3.4.65 |
| reference_document_identifier (attribute of *Reference_Document)* | 5.1.13.2.1 |
| reference document language identifier | 3.4.66 |
| reference_document_language_identifier (attribute of *Reference_Document)* | 5.1.13.2.3 |
| reference document notation | 3.4.67 |
| reference_document_notation (attribute of *Reference_Document)* | 5.1.13.2.4 |
| reference document title | 3.4.68 |
| reference_document_title (attribute of *Reference_Document)* | 5.1.13.2.4 |
| reference document type description | 3.4.69 |
| reference_document_type_description (attribute of *Reference_Document)* | 5.1.13.2.2 |
| reference document uri | 3.4.70 |
| reference_document_uri (attribute of *Reference_Document)* | 5.1.13.2.7 |
| reference provider | 3.4.71 |
| reference_provider (attribute of *Reference_Document)* | 5.1.13.2.6 |
| reference_type (attribute of *Reference)* | 7.1.5.2.2.1 |
| reflexive relation | 3.4.72 |
| reflexivity (attribute of *Binary_Relation)* | 9.1.2.2.2.1 |
| Reflexivity (enumeration) | 9.1.2.3 |
| region_identifier | 5.1.7.2.3 |
| register | 3.4.73 |
| registered item | 3.4.74 |
| Registered_Item (type) | 7.1.2.1 |
| registrar | 3.4.75 |
| Registrar (class) | 7.1.2.4 |
| registrar identifier | 3.4.76 |
| registrar_identifier (Attribute of *Registrar*) | 7.1.2.4.2.1 |
| registration | 3.3.32 |
| Registration (association class) | 7.1.5.1 |
| registration authority (RA) | 3.4.77 |

| Term | Defined in |
|---|---|
| Registration_Authority (class) | 7.1.2.5 |
| registration authority identifier | 3.4.78 |
| Registration_Authority_Identifier (datatype, class) | 5.1.14 |
| registration_authority_identifier_space (association) | 7.1.6.2 |
| registration_authority_registrar (association) | 7.1.6.3 |
| registration record | 3.4.79 |
| Registration_Record (class, datatype) | 7.1.2.6 |
| registration_state (Attribute of *Registration_Record*) | 7.1.5.1.2.1 |
| registration status | 3.4.80 |
| registration_status (Attribute of *Registration_Record*) | 7.1.2.6.2.1 |
| registry_character_repertoire (Attribute of *Registry_Specification*) | 7.1.2.9.2.5 |
| registry_comment (Attribute of *Registry_Specification*) | 7.1.2.9.2.10 |
| registry_conformance_level (Attribute of *Registry_Specification*) | 7.1.2.9.2.4 |
| registry_context (Attribute of *Registry_Specification*) | 7.1.2.9.2.9 |
| registry instance | 3.3.33 |
| registry item | 3.3.34 |
| registry metamodel | 3.3.35 |
| registry_name (Attribute of *Registry_Specification*) | 7.1.2.9.2.1 |
| registry_primary_language (Attribute of *Registry_Specification*) | 7.1.2.9.2.7 |
| registry product | 3.3.36 |
| registry_reference_document_identifier _form (Attribute of *Registry_Specification*) | 7.1.2.9.2.6 |
| registry_representation_class_scheme (Attribute of *Registry_Specification*) | 7.1.2.9.2.8 |
| registry_standard (Attribute of *Registry_Specification*) | 7.1.2.9.2.3 |
| registry_web_address (Attribute of *Registry_Specification*) | 7.1.2.9.2.2 |
| Registry_Specification (class) | 7.1.2.9 |
| related metadata reference | 3.3.37 |
| relation | 3.4.81 |
| Relation (class) | 8.1.2.4, 9.1.2.1 |
| relation_membership (association) | 8.1.3.7 |
| Relation_Role (class) | 8.1.2.5 |
| relation_role_set (association) | 8.1.3.8 |

| Term | Defined in |
|---|---|
| relationship <metamodel> | 3.2.11 |
| scope_rule (Attribute of *Naming_Convention*) | 6.2.2.7.2.1 |
| scoped identifier | 3.4.82 |
| Scoped_Identifier (class) | 6.1.2.2 |
| script_identifier (Attribute of *Language_Identification*) | 5.1.7.2.2 |
| semantic_rule (Attribute of *Naming_Convention*) | 6.2.2.7.2.3 |
| shorthand_expansion (Attribute of *Scoped_Identifier*) | 6.1.2.2.2.3 |
| shorthand_prefix (Attribute of *Namespace*) | 6.1.2.3.2.5 |
| sign (noun) | 3.4.83 |
| Sign (class, datatype) | 5.1.15 |
| SKOS | 3.5.11 |
| slot | 3.4.84 |
| Slot (class) | 6.1.2.4 |
| slot_name (Attribute of *Slot*) | 6.1.2.4.2.1 |
| slot_type (Attribute of *Slot*) | 6.1.2.4.2.3 |
| slot_value (Attribute of *Slot*) | 6.1.2.4.2.2 |
| steward | 3.4.85 |
| steward (Attribute of *Stewardship_Record)* | 7.1.2.7.2.1 |
| stewardship (of Administered_Item) | 7.1.6.4 |
| stewardship (of metadata) | 3.3.38 |
| stewardship contact | 3.4.86 |
| stewardship_contact(Attribute of *Stewardship_Record)* | 7.1.2.7.2.2 |
| stewardship record | 3.4.87 |
| Stewardship_Record | 7.1.2.7 |
| string | 3.4.88 |
| String (datatype, class) | 5.1.16 |
| submission | 3.4.89 |
| submission (association) | 7.1.6.5 |
| submission contact | 3.4.90 |
| submission_contact (Attribute of *Submission_Record*) | 7.1.2.8.2.2 |
| submission record | 3.4.91 |
| Submission_Record (class) | 7.1.2.8 |
| submitter (Attribute of *Submission_Record*) | 7.1.2.8.2.1 |
| symmetric relation | 3.4.92 |
| symmetry (Attribute of *Binary_Relation*) | 9.1.2.2.2.2 |

| Term | Defined in |
|---|---|
| Symmetry (enumeration) | 9.1.2.4 |
| syntactic_rule<br>(Attribute of *Naming_Convention*) | 6.2.2.7.2.4 |
| term_definition_pairing | 6.2.4.6 |
| text | 3.3.39 |
| Text (datatype) | 5.1.17 |
| transitive relation | 3.4.93 |
| transitivity<br>(Attribute of *Binary_Relation*) | 9.1.2.2.2.3 |
| Transitivity (enumeration) | 9.1.2.5 |
| UML | 3.5.12 |
| unit of measure | 3.4.94 |
| Unit_of_Measure (class) | 10.4.2.1 |
| unit_of_measure_dimensionality<br>(association) | 10.4.3.1 |
| unresolved_issue<br>(attribute of *Registration_Record*) | 7.1.2.6.2.5 |
| until_date<br>(attribute of *Registration_Record*) | 7.1.2.6.2.3 |
| UPU | 3.5.13 |
| Value (class) | 5.1.18 |
| value domain (VD) | 3.4.96 |
| Value_Domain (class) | 10.3.2.5 |
| value_domain_datatype<br>(attribute of *Value_Domain*) | 10.3.2.5.2.1 |
| value_domain_description<br>(attribute of *Described_Value_Domain*) | 10.3.2.8.2.1 |
| value_domain_format<br>(attribute of *Value_Domain*) | 10.3.2.5.2.2 |
| value_domain_maximum_character_quantity<br>(attribute of *Value_Domain*) | 10.3.2.5.2.3 |
| value_domain_meaning (association) | 10.3.3.1 |
| value_domain_unit_of_measure<br>(attribute of *Value_Domain*) | 10.3.2.5.2.4 |
| value meaning | 3.4.97 |
| Value_Meaning (class) | 10.3.2.3 |
| value_meaning_begin_date<br>(attribute of *Value_Meaning*) | 10.3.2.3.2.1 |
| value_meaning_end_date<br>(attribute of *Value_Meaning*) | 10.3.2.3.2.2 |
| value_meaning_set (association) | 10.3.3.2 |
| variant_identifier<br>(attribute of *Language_Identification*) | 5.1.7.2.4 |
| VD | 3.5.14 |
| version | 3.4.98 |
| version<br>(attribute of *Administered_Item*) | 7.1.2.2.2.1 |

| Term | Defined in |
|---|---|
| W3C | 3.5.15 |
| XCL | 3.5.16 |
| XML | 3.5.17 |
| XTML | 3.5.18 |

# Annex B
(normative)

# Consolidated Class Hierarchy

Figure B-1 shows all classes that participate in a class hierarchy. Classes that do not participate in a hierarchy are not shown.

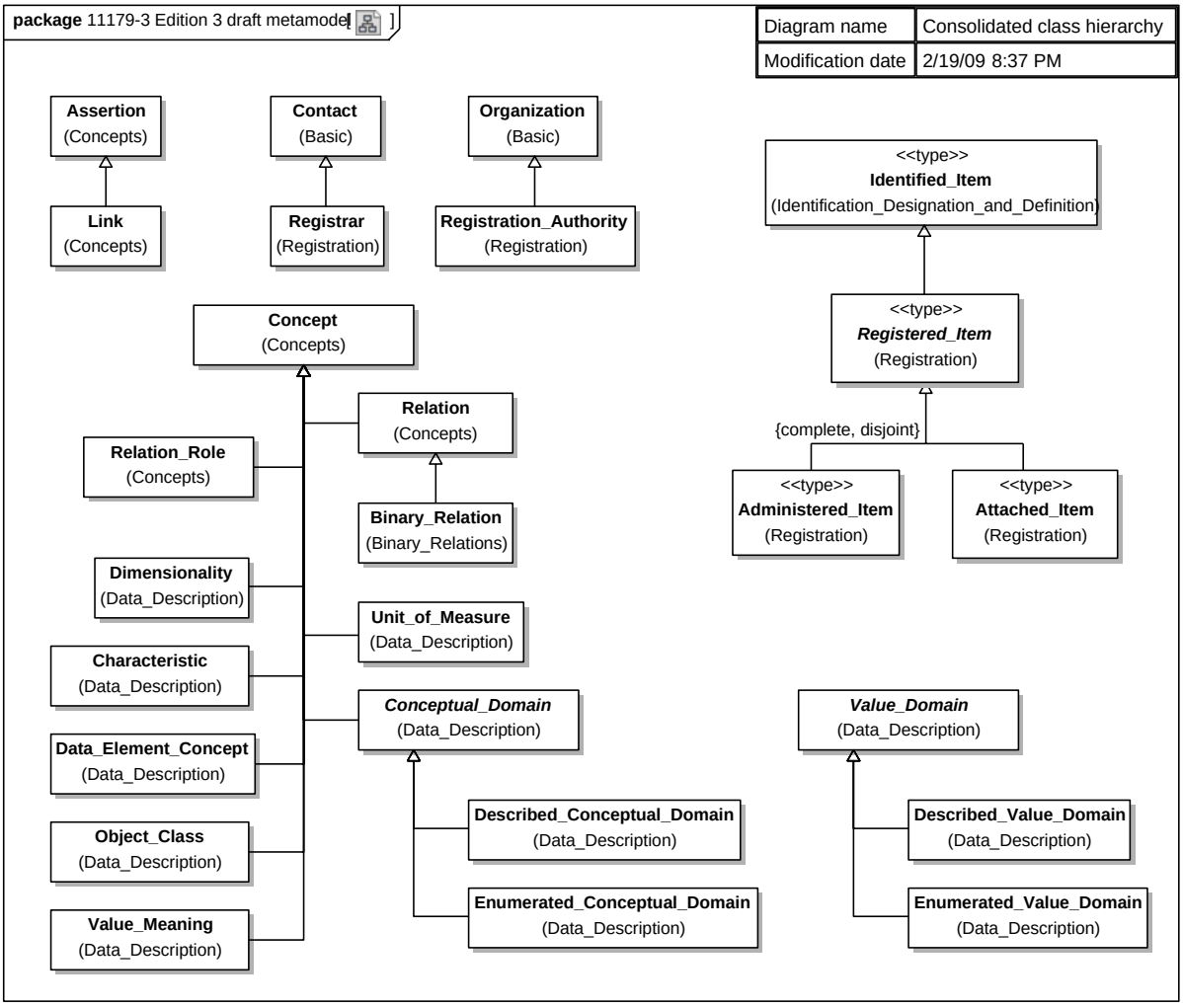



**Figure B-1 — Consolidated Class Hierarchy**

# Annex C
(informative)

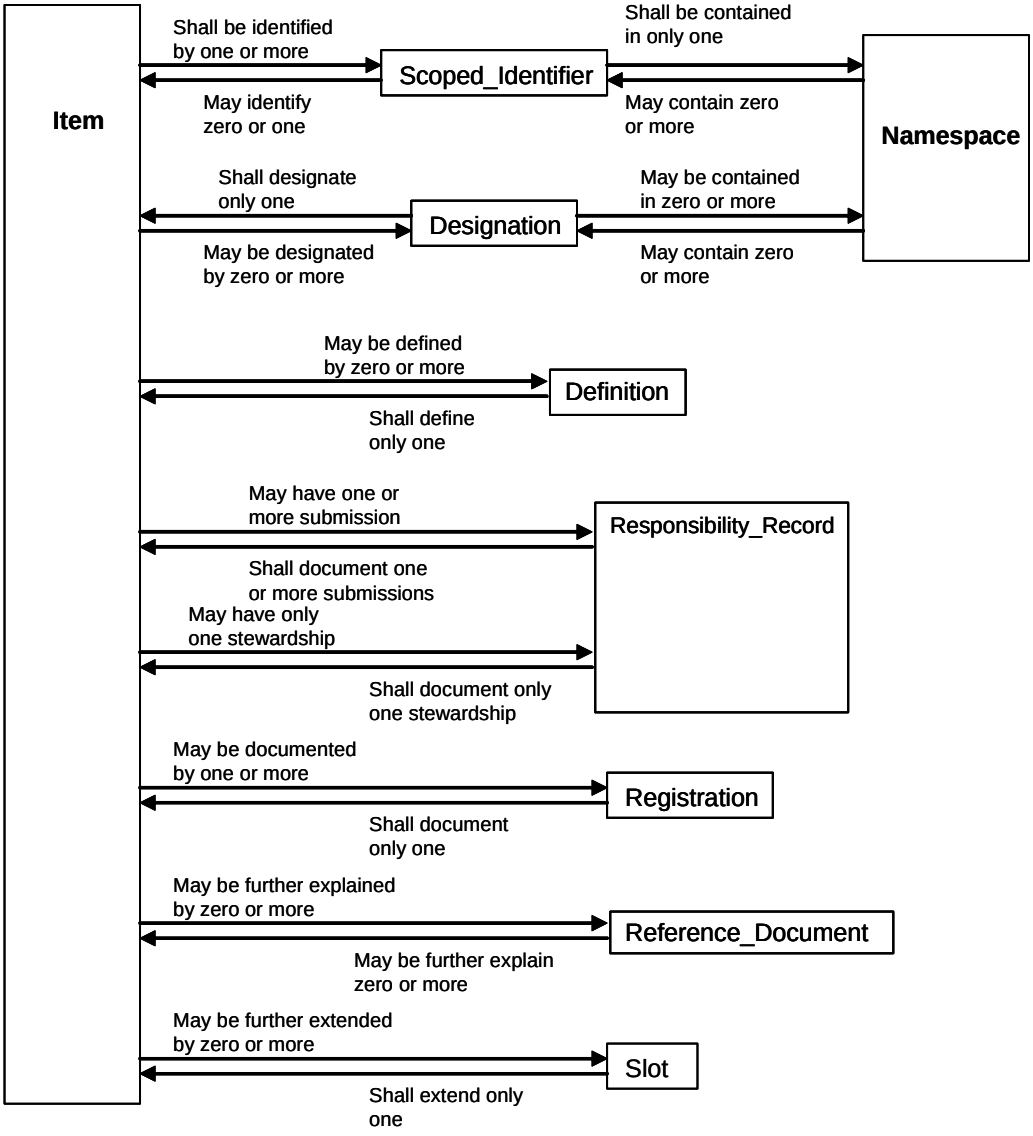
# Alternative representation of the metamodel

EDITOR'S NOTE #48. '(Action required) This Annex was added by US comment 44 on the CD1 ballot. No text was provided with the figures, and no explanation of the rationale for the Annex, nor explanation of the notation was provided. The US is requested to complete this Annex, or it will be removed for the FCD ballot. There are also typos in some of the Figures, and discrepancies between this model and the UML model which need to be addressed.

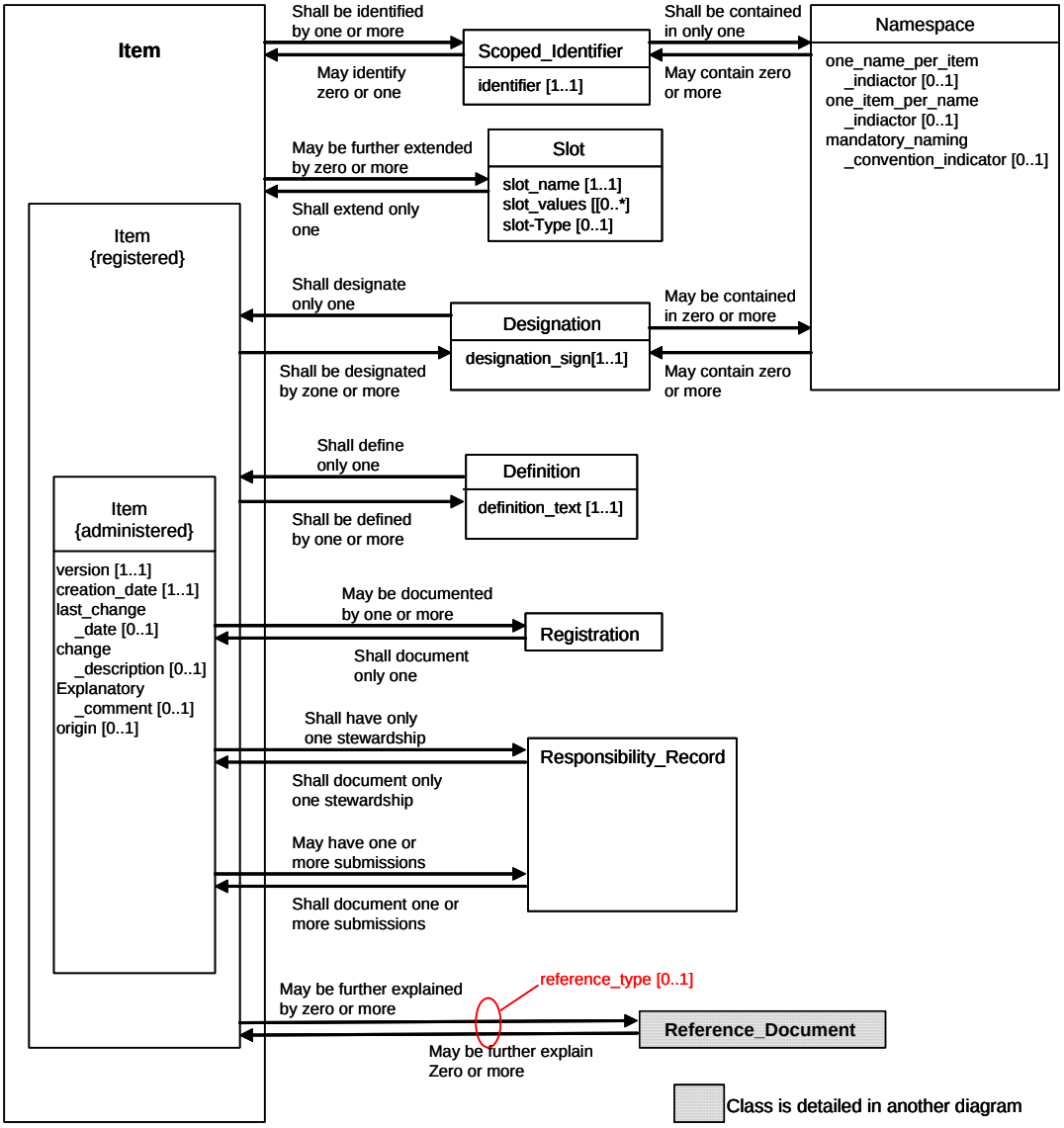
EDITOR'S NOTE #49. '(Action required) Add Captions to the Figures.

This Annex attempts to present a more intuitive representation of the metamodel.

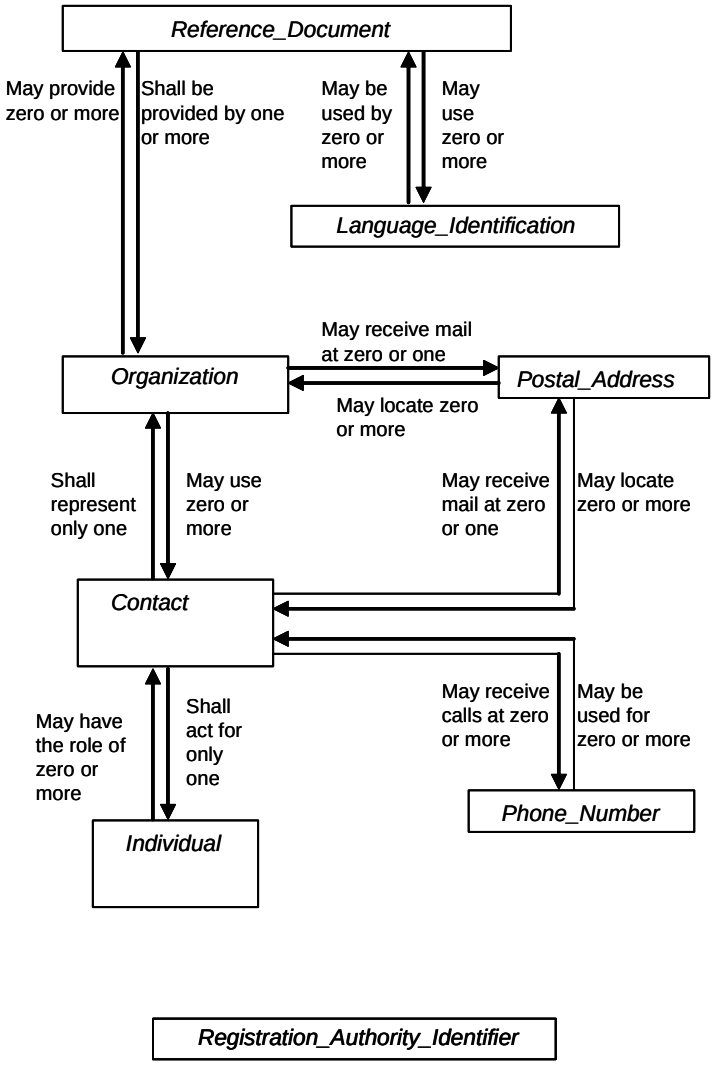
## C.1 Major Item Elements



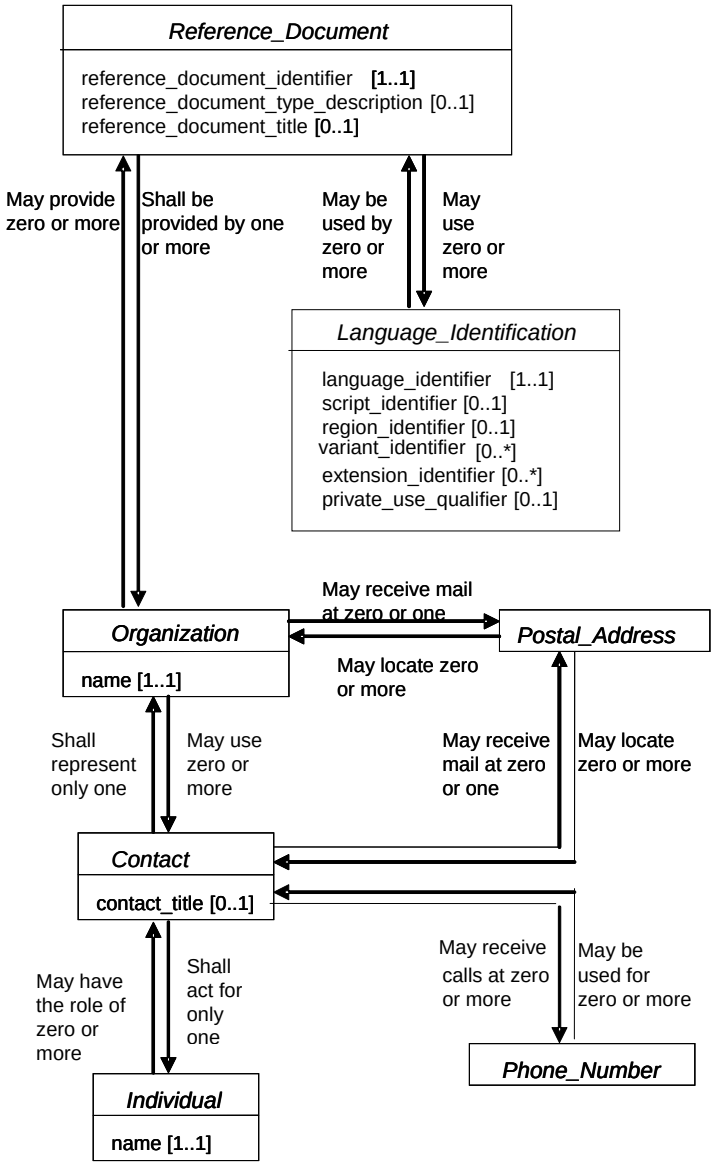
## C.2 Major Item Elements Detail

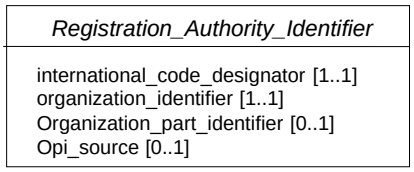


## C.3 Supporting Elements

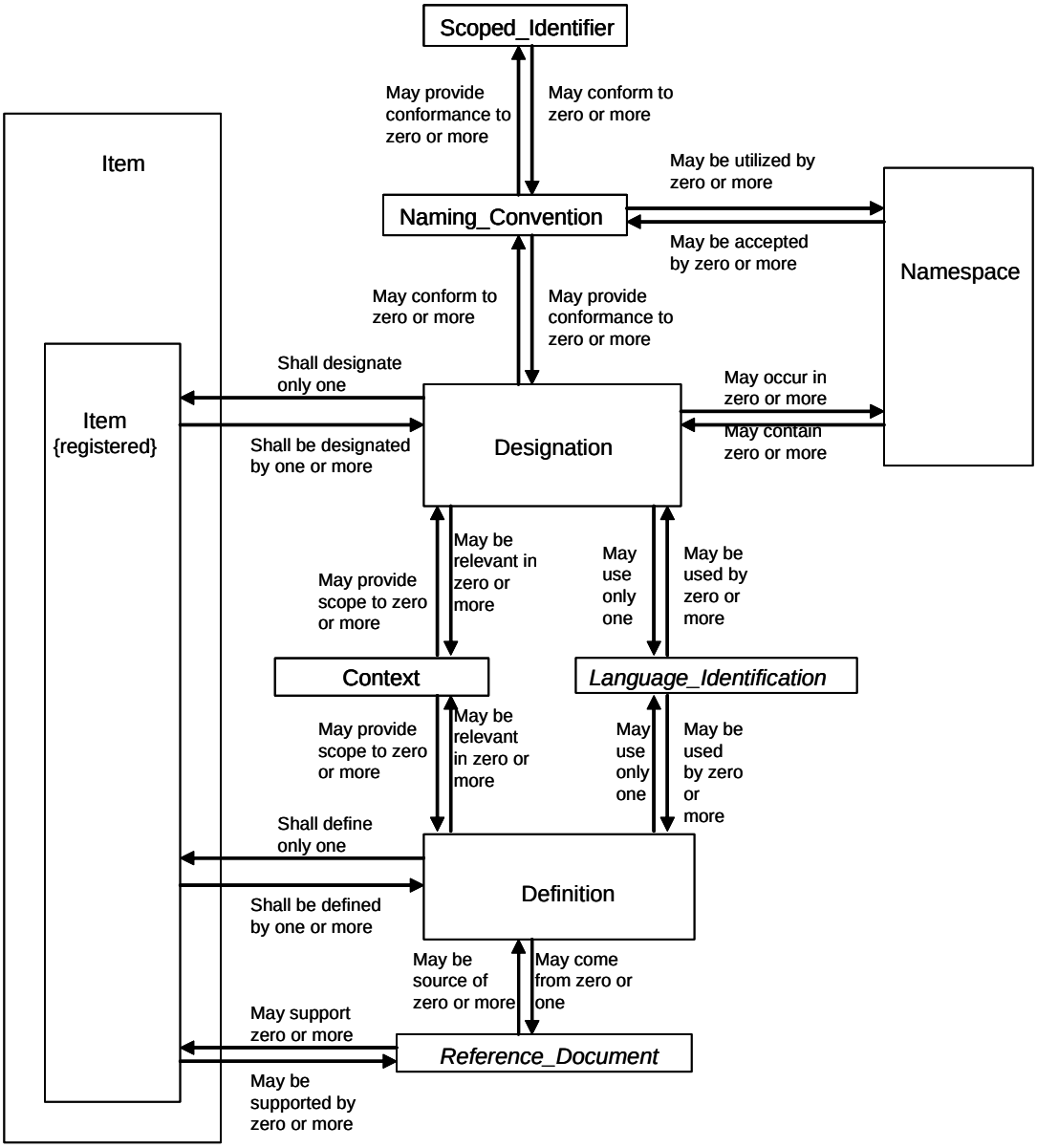


## C.4 Supporting Elements Detail

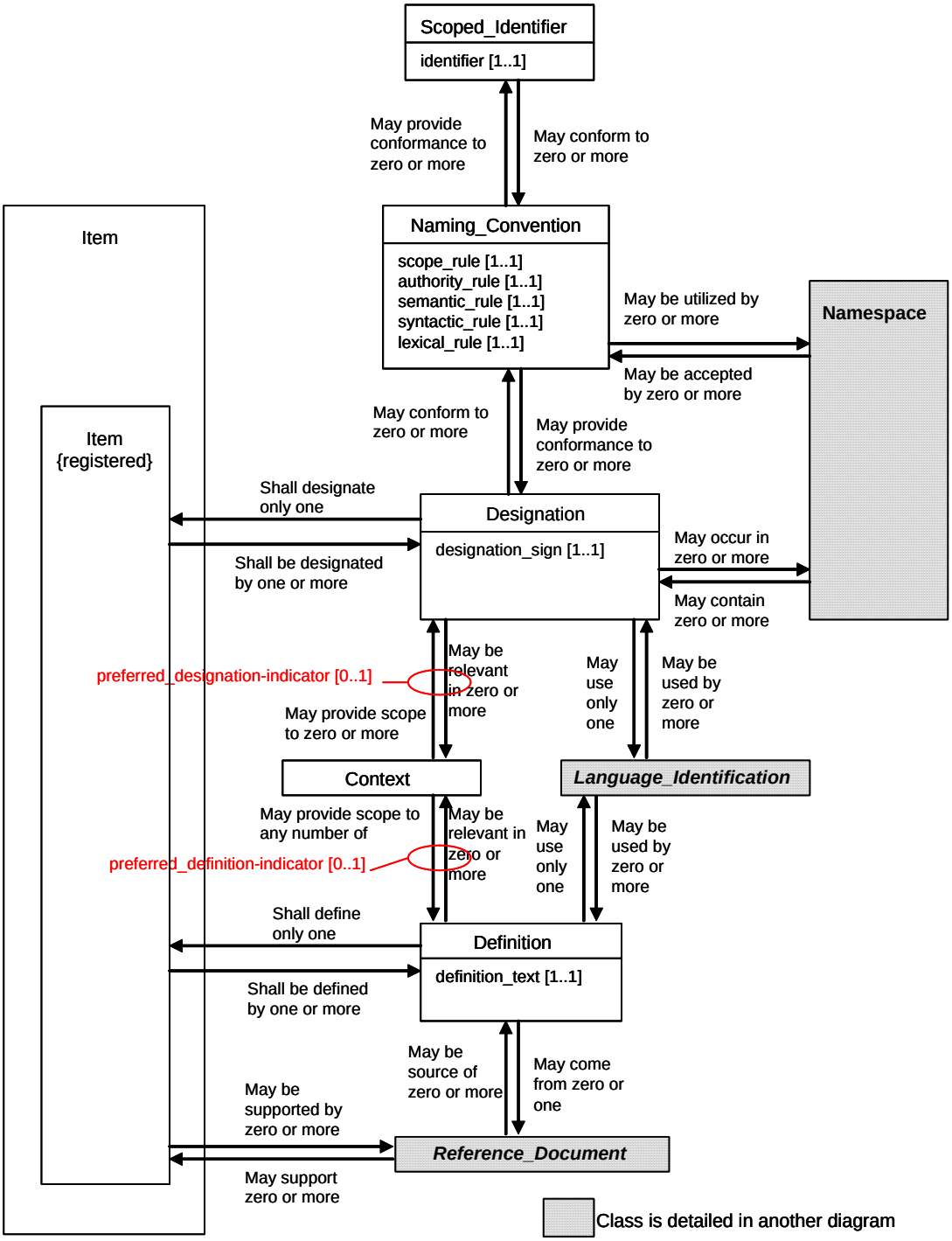


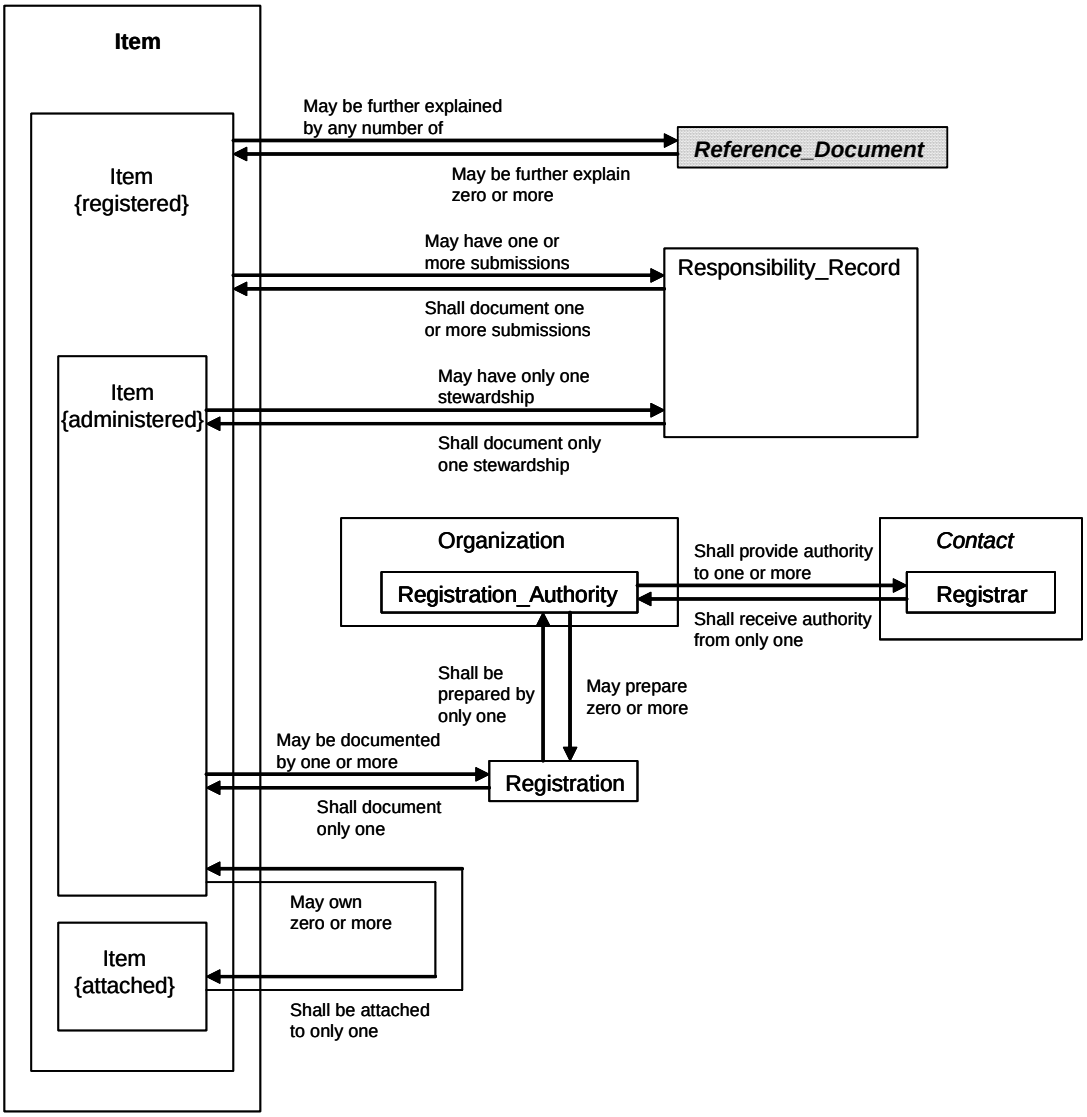
## C.5 Definition & Designation Elements



## C.6 Definition & Designation Elements Detail



## C.7 Item Registration Elements

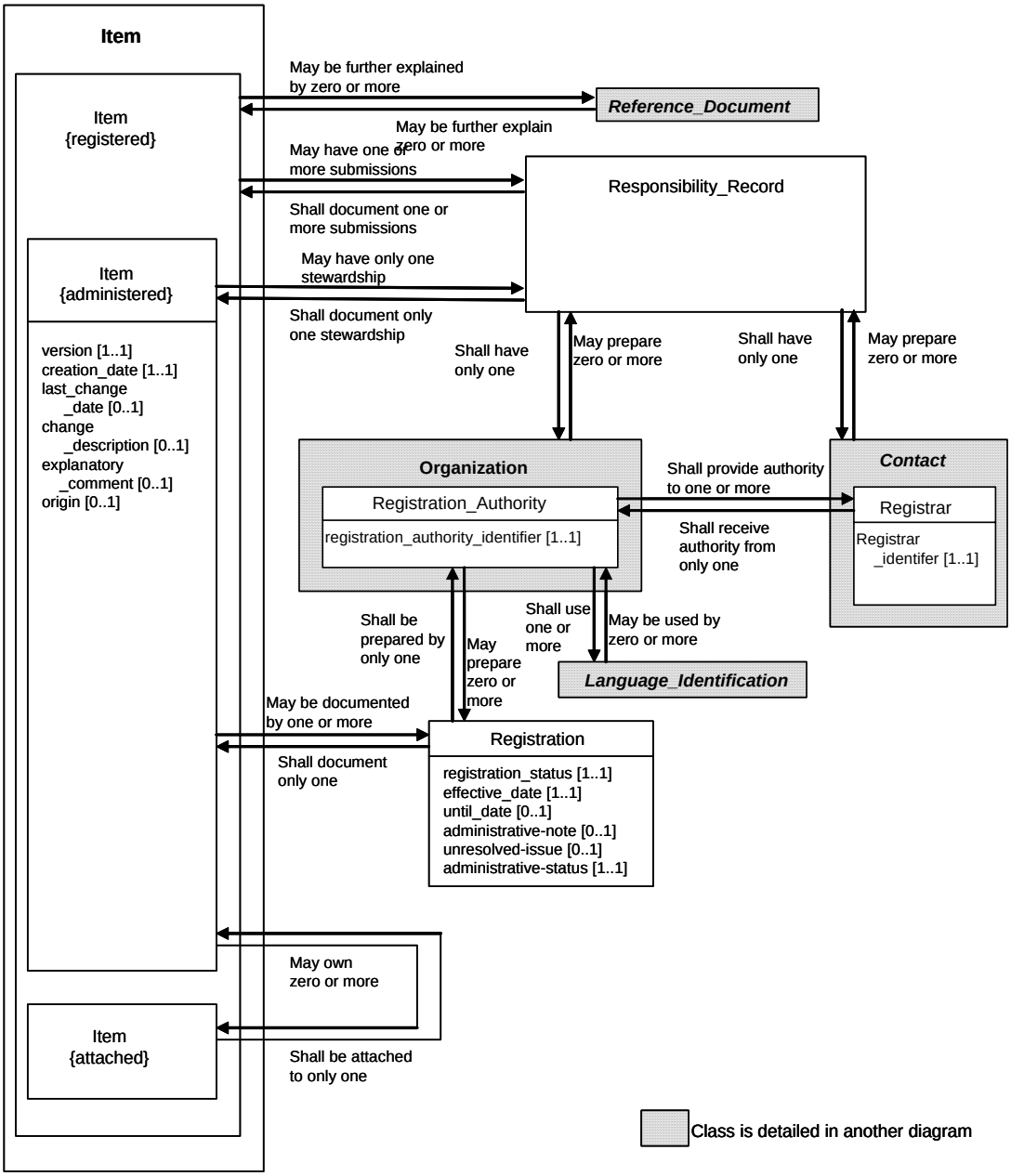


## C.8 Item Rules

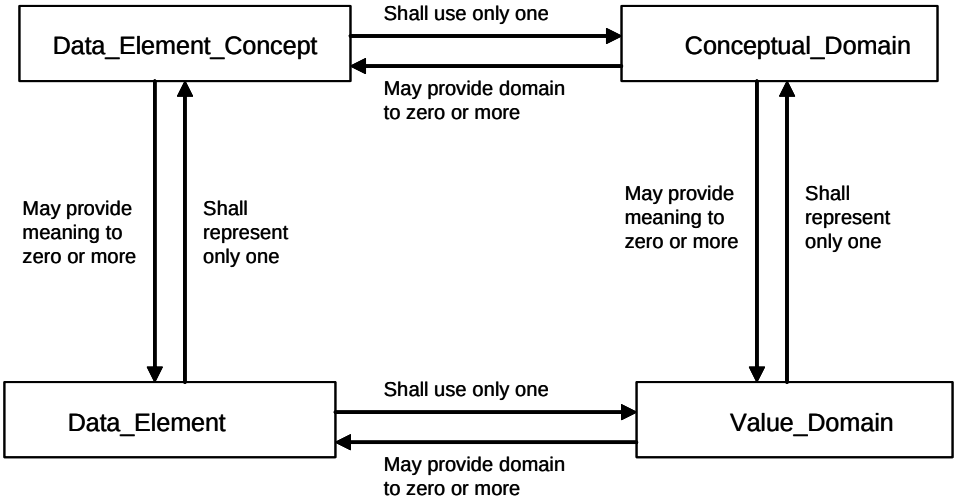
The following decision table describes the rules that apply for items to be: identified, registered, administered or attached.

| | Rule 1 | Rule 2 | Rule 3 | Rule 4 |
|---|---|---|---|---|
| Item instance has instance of: | | | | |
| Scoped_Identifier | one or more | one or more | one or more | one or more |
| Designation | | one or more | one or more | one or more |
| Definition | | one or more | one or more | one or more |
| Submission_Record | | one or more | one or more | one or more |
| Stewardship_Record | | | one | no |
| Registration | | | one or more | no |
| attachment | | | | one |
| | | | | |
| Item is identified | **X** | | | |
| Item is registered | | **X** | | |
| Item is administered | | | **X** | |
| Item is attached | | | | **X** |

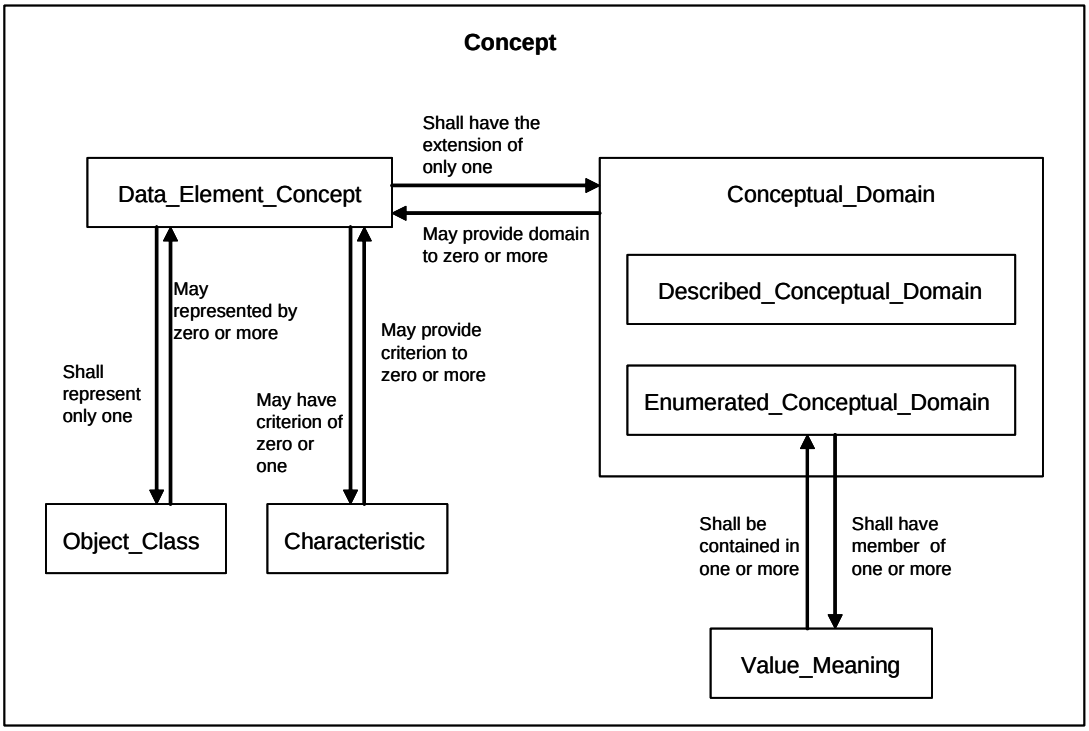
## C.9 Item Registration Elements Detail



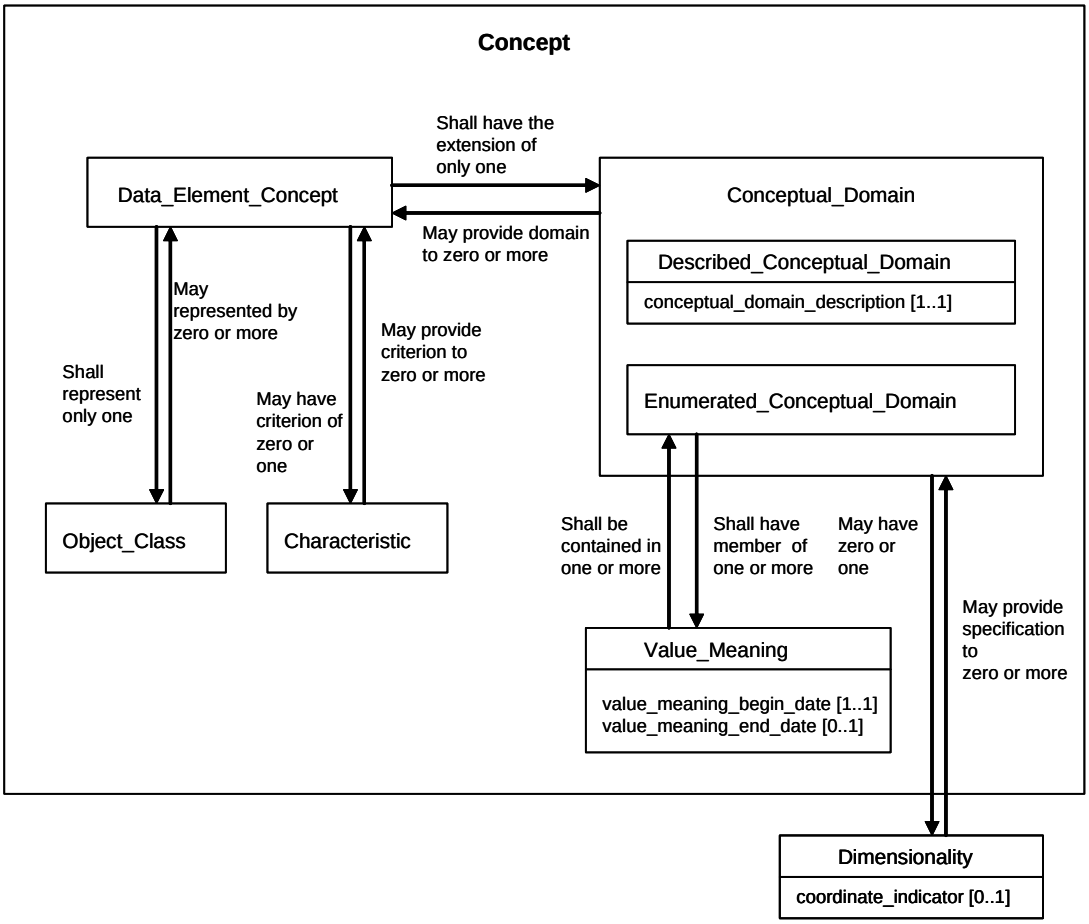
## C.10 Data Description



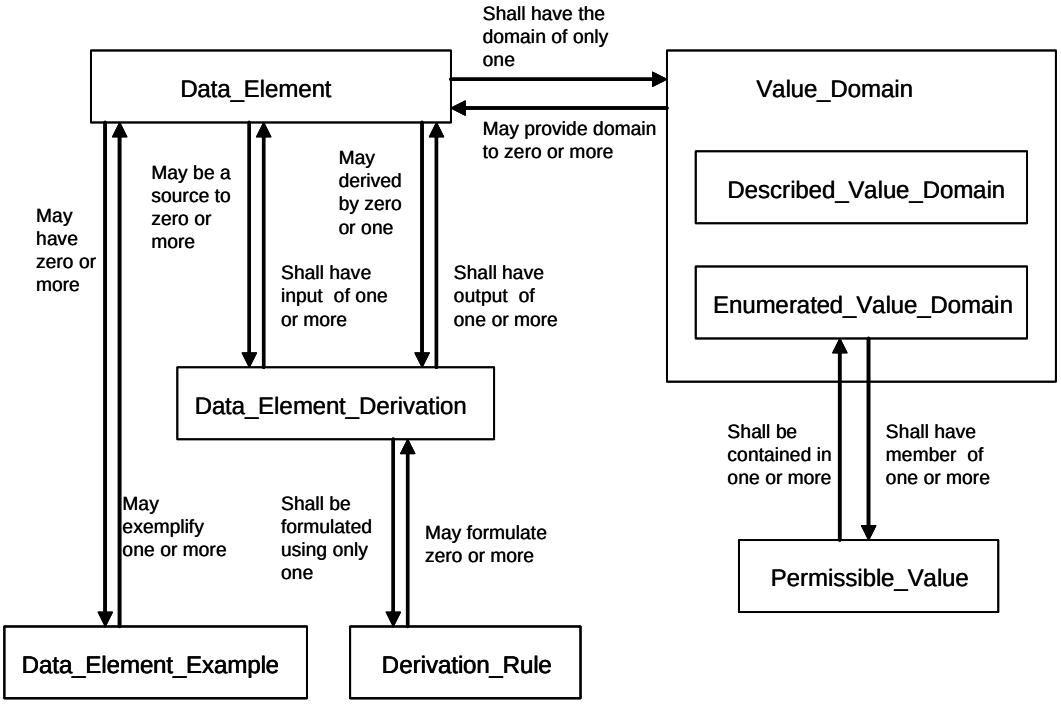
## C.11 Concept



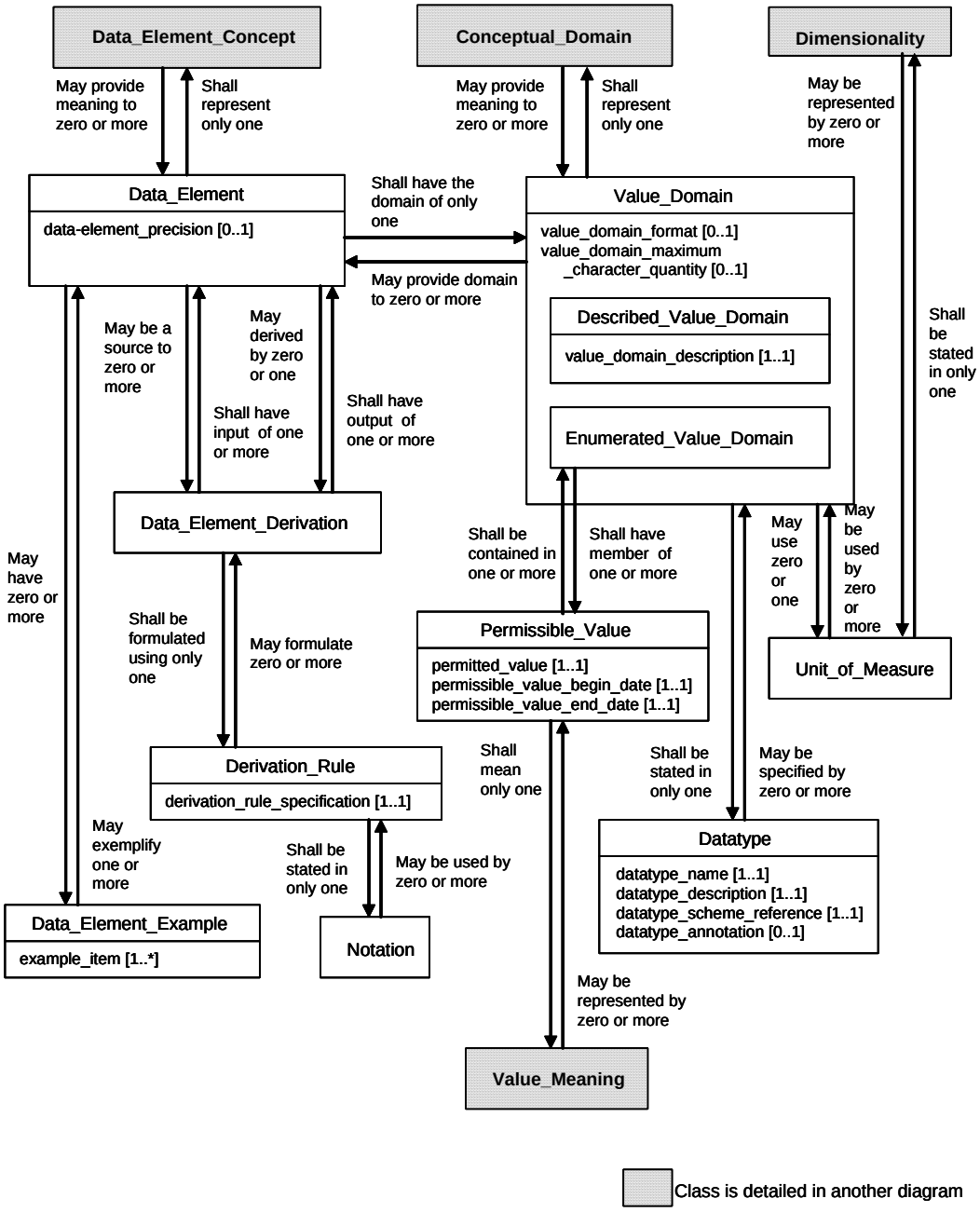
## C.12 Concept Detail



## C.13 Data Element & Value Domain



## C.14 Data Element & Value Domain Detail



# Annex D
(informative)

# Mapping the ISO/IEC 11179-3:1994 basic attributes to the ISO/IEC 11179-3:200n metamodel and basic attributes

EDITOR'S NOTE #50. (Action required) In this edition we need to map from both Edition 1 and Edition 2 to Edition 3. This Annex has not yet been revised from Edition 2. It will be revised for the FCD once the model has stabilized.

EDITOR'S NOTE #51. (Action required) In accordance with ISO Directives, all tables in this Annex should be given an id and a name and included in the Table of Tables at the front of the document. To be done for FCD after the Annex has been revised.

## D.1 Introduction

ISO/IEC 11179-3:1994 lists 23 basic attributes of data elements, as shown in Figure D.1.

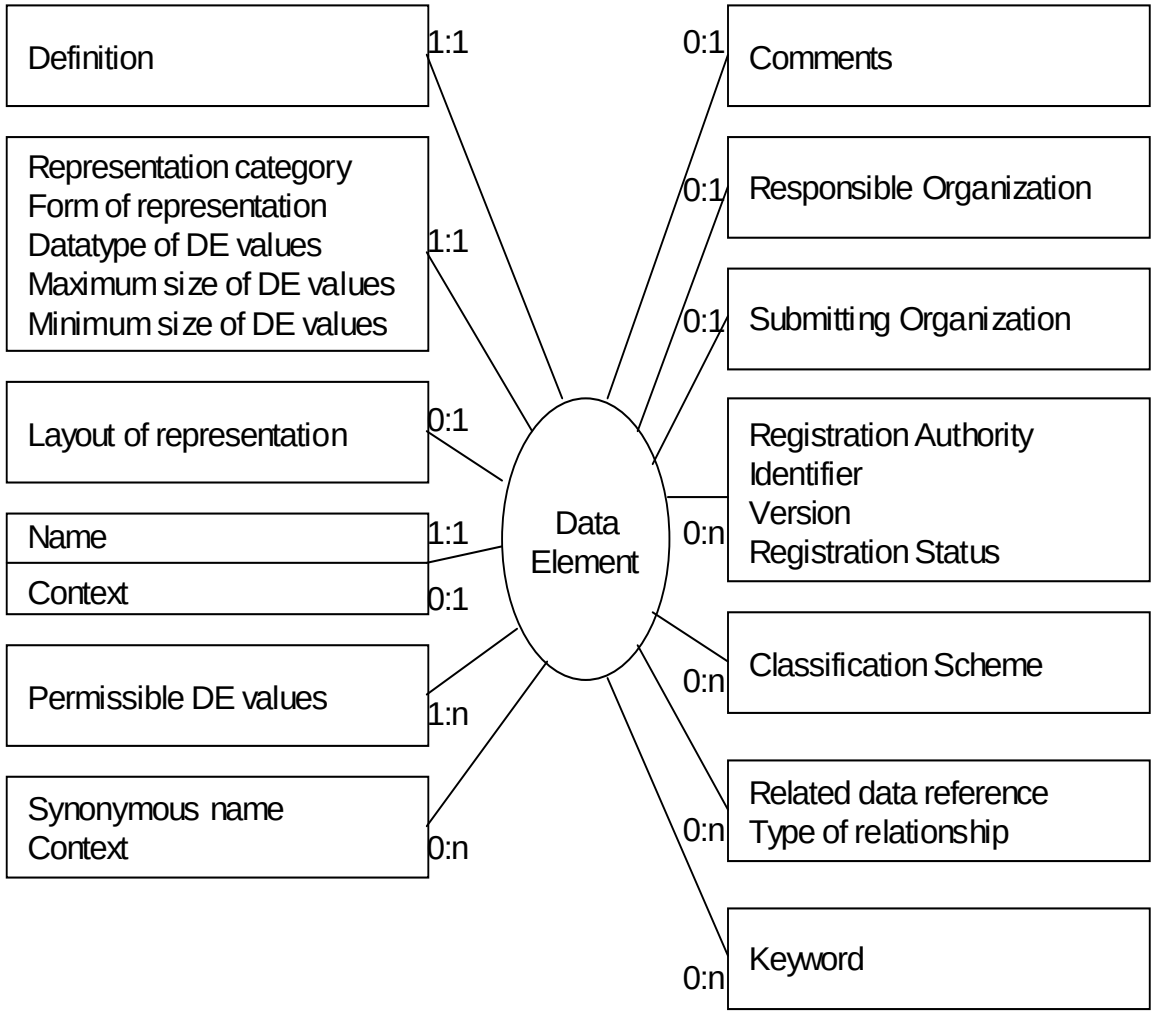



**Figure D.-1 — Basic Attributes of Data elements**

This edition of the standard supports not only data elements, but also other metadata items associated with them, such as data element concepts, conceptual domains and value domains.

This annex maps the 1994 basic attributes to the metamodel in Clauses 4 through 10, and the new basic attributes in Clause 11.

### D.1.1 Description of Table Structures in this Annex

EDITOR'S NOTE #52. (Action required) The Editor proposes to use the following column headings when this Annex is revised:
Edition 1 Basic Attributes; Edition 2 Basic Attributes; Edition 3 Basic Attributes;
Edition 2 Model; Edition 3 Model.

The tables in this Annex are structured as follows:

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | | | |
| Definition: | | | |
| Obligation: | | | |
| Condition: | | | |
| Datatype: | | | |
| Comment: | | | |

Path from Administered_Item:

#### D.1.1.1 Description of the Columns

The columns in the table are used as follows:

— Column 1: Label for the row

— Column 2: What was specified in ISO/IEC 11179-3:1994 Clause 6

— Column 3: What is specified in ISO/IEC 11179-3:2002 Clause 4

— Column 4: What is specified in ISO/IEC 11179-3:2002 Clause 5

#### D.1.1.2 Description of the Rows

The rows in the table are used as follows, with the value in a particular cell coming from the Clause identified by the column (see above).

NOTE For the purposes of reference in the following text, the rows are numbered beginning at 1, and ignoring the column headings.

— Row 1: Attribute name - Contains the name of the attribute. For column 3, this is specified as: “Class name” “attribute name”, where “Class name” designates the Class in the metamodel that contains the attribute.

— Row 2: Definition – Contains the definition of the attribute.

— Row 3: Obligation – Contains the obligation of the attribute. (One of: Mandatory, Optional or Conditional.)

— Row 4: Condition – If the Obligation is “Conditional”, this row contains the condition that applies. (The entire row is omitted if it is not relevant for any column.)

— Row 5: Datatype – Contains the datatype of the attribute.

— Row 6: Comment – Contains any explanatory_comment. (The entire row is omitted if it is not relevant for any column.)

The notation "N/A" indicates that a row is "Not Applicable" for a particular column.

#### D.1.1.3 Specification of attribute name in row 1 column 3

For the old and new basic attributes (columns 2 and 4 respectively) the attribute name is straightforward. The equivalent attributes in the metamodel (column 3), need to be designated in the context of a particular class. The class that provides the context is named first, and then the attribute, using the "dot" notation:

"Class Name" . "attribute name"

e.g. “Item Identifier” . “version”

#### D.1.1.4 Specification of Path from Administered_Item to the named attribute

This information shows how the named attributed is related to an Administered_Item, and applies to column 3 only. It has been placed after the table to save space, and make the path easier to read. It specifies the path that needs to be navigated in the metamodel to reach the named attribute for any particular Administered_Item. (See below for an explanation of the notation.) Whenever the attribute is on the Administered_Item class, no navigation is necessary and this row is omitted.

In addition to designating the metamodel attribute in the context of a class (row 1 column 3), the “Path to Administered_Item” shows how the class is related an Administered_Item. It is necessary to navigate relationships and/or composite attributes within the model from one class to another. For common attributes (i.e. those that apply to any Administered_Item), the starting point for navigation is the supertype class “Administered_Item”. For attributes specific to a particular subtype of Administered_Item, the starting point for navigation is that subtype class (e.g. Data_Element). The “dot” notation is used as described below.

NOTE 1 The following notational convention is used:

— the names of classes and composite datatypes are capitalized e.g. "Item Identifier"

— the names of attributes are all lower case e.g. “version”

— the names of relationships are lower case and italicised e.g. “*name entry*”

NOTE 2 The use of *italics* to indicate a relationship applies only to the specification of the navigation path. In row 2 of the table (Definition), *italics* are used to distinguish the term from the definition.

Example 1: Attribute “version”

In this example, the attribute is a Common Attribute (i.e. it can apply to any type of Administered_Item), so the navigation starts from the supertype class “Administered_Item”.

“Administered_Item”. “administered item administration record” .
“Administration_Record”. “administered item identifier” .
“Item Identifier”. “version”

specifies to follow the path in the model:

— from the class “Administered_Item” via its attribute “administered item administration record” to the composite datatype “Administration_Record”, then

— from the class “Administration_Record” via its attribute “administered item identifier” to the composite datatype “Item Identifier” and its attribute “version”.

**Example 2: Attribute "datatype_name"**

In this example, the attribute is specific to a Data_Element, so the navigation starts from the “Data_Element” subtype class of Administered_Item.

“Data_Element”. “*data element representation*”.
“Value_Domain”. “value_domain_datatype”.
“Datatype”. “datatype_name”

specifies to follow the path in the model:

— from the class “Data_Element” via its relationship “*data element representation*” to the related class “Value_Domain”, then

— from the class “Value_Domain” via its attribute “value_domain_datatype” to the composite datatype “Datatype” and its attribute “datatype_name”.

## D.2 Mapping the Basic Attributes

The attributes are ordered in this Annex as in Clause 5 of this document.

### D.2.1 Common Identifying attributes

#### D.2.1.1 Name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Name | “Designation (of Administered_Item)”. “name” | name |
| Definition: | Single or multi word designation assigned to a data element. | A name by which an Administered_Item is known within a specific Context. | A name by which a metadata item is known within a specific context. |
| Obligation: | Mandatory | Mandatory | Mandatory |
| Data type: | Character string | String | String |
| Comment: | | The attribute “preferred designation” may be used to specify the primary name if synonyms also exist in a particular context. | |

##### D.2.1.1.1 Path from Administered_Item:

“Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “*name entry*”.

#### D.2.1.2 Synonymous name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Synonymous name | “Designation (of Administered_Item)”. “name” | name |
| Definition: | Single word or multi word designation that differs from the given name, but represents the same data element concept. | A name by which an Administered_Item is known within a specific Context. | A name by which a metadata item is known within a specific context. |
| Obligation: | Optional | Optional | Optional |
| Data type: | Character string | String | String |
| Comment: | Synonymous names are often familiar names in a certain application environment. If this is the case use attribute 'Context' (6.1.6) to specify the context. If more synonymous names occur the attributes 'Synonymous name' and 'Context' shall be specified as a pair. | An Administered_Item may have multiple names in the same or different contexts. The distinction between “name” and “synonymous name” in a particular context may be specified by the attribute “preferred designation”, which should be set to “True” for the preferred name, and “False” for all synonyms. | A metadata item may have multiple names in the same or different contexts. The distinction between “name” and “synonymous name” in a particular context may be specified by the attribute “preferred designation”, which should be set to “True” for the preferred name, and “False” for all synonyms. |

##### D.2.1.2.1 Path from Administered_Item:

“Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “*name entry*”.

#### D.2.1.3 Context name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Context | “Designation (of Administered_Item)” . “name”<br>NOTE The "Administered_Item" referred to here is the Context itself, not the Administered_Item to which context is being provided. | context name |
| Definition: | A designation or description of the application environment or discipline in which a name and/or synonymous name is applied or originates from.<br>Note: The latest edition of the standard differentiates designations from descriptions. | *Context:* A universe of discourse in which a name or definition is used.<br>*name:* A name by which an Administered_Item (in this case the Context) is known within a specific context (where the context for a Context is probably the registry). | *Context:* A universe of discourse in which a name or definition is used.<br>*name:* A name by which a metadata item (in this case the Context) is known within a specific context (where the context for a context is the setting in which it is used). |
| Obligation: | Conditional | Mandatory | Conditional |
| Condition: | This attribute is mandatory for each occurrence of the attribute 'Synonymous name' (6.1.5). This attribute is mandatory when the attribute | N/A | Required if more than one *name* attribute exists for a particular metadata item. |

| | | | |
|---|---|---|---|
| | 'Name' (6.1.1) occurs in an information exchange. | | |
| Data type: | Character string | String | String |
| Comment: | Assignment of the attribute 'Context' to the attribute 'Name' may be made mandatory as part of the procedures of any Registration_Authority. | As an Administered_Item itself, any Context used within a registry must be given both a *name* and *definition*. A Context must itself exist within a Context, which for most will probably be the registry. (A Context may provide Context to itself.) | |

#### D.2.1.3.1 Path from Administered_Item:

“Administered_Item”(1). “*administered item context*”.“Context”. “context administration record”. “Administration_Record” . “Administered_Item” (2). “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “name entry”.

NOTES (1) (2) The first “Administered_Item” is the one to which context is being provided. The second "Administered_Item" is the Context itself.

#### D.2.1.4 Context identifier

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Context". "context administration record". "Administration_Record". "administered item identifier" | context identifier |
| Definition: | N/A | *Context:* A universe of discourse in which a name or definition is used.<br>*administered item identifier:* The unique identifier for an Administered_Item (in this case the Context). | *Context:* A universe of discourse in which a name or definition is used.<br>*context identifier:* A unique identifier for the *Context* within its usage context. |
| Obligation: | N/A | Mandatory | Conditional |
| Condition: | N/A | N/A | Required if *context name* is not unique with its usage context. |
| Data type: | N/A | String | String |
| Comment: | | As an Administered_Item itself, any Context used within a registry must be given an *administered item identifier*. | |

##### D.2.1.4.1 Path from Administered_Item:

"Administered_Item". "*administered item context*".

#### D.2.1.5 Context description

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Context | "Context". "context description" | context description |
| Definition: | A designation or description of the application environment or discipline in which a name and/or synonymous name is applied or originates from.<br>Note: The new metamodel differentiates designations from descriptions. | *Context:* A universe of discourse in which a name or definition is used.<br>*context description:* The textual description of the context. | *Context:* A universe of discourse in which a name or definition is used.<br>*context description:* The textual description of the context. |
| Obligation: | Conditional | Mandatory | Conditional |
| Condition: | This attribute is mandatory for each occurrence of the attribute 'Synonymous name'. This attribute is mandatory when the attribute 'Name' occurs in an information exchange. | N/A | Required if *context name* is used. |
| Data type: | Character string | String | String |
| Comment: | Assignment of the attribute 'Context' to the attribute 'Name' may be made mandatory as part of the procedures of any Registration_Authority. | In this edition of this part of ISO/IEC 11179, *context description* and *context name* exist as two separate attributes. | In this edition of this part of ISO/IEC 11179, *context description* and *context name* exist as two separate attributes. |

##### D.2.1.5.1 Path from Administered_Item:

"Administered_Item". "*administered item context*".

#### D.2.1.6 Item identifier – data identifier

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Identifier | "Item Identifier" . "data identifier" | item identifier – data identifier |
| Definition: | A language independent unique identifier of a data element within a Registration_Authority. | The unique identifier for an Administered_Item within a Registration_Authority. | The unique identifier for a metadata item within a specific context. |
| Obligation: | Conditional | Mandatory | Conditional |
| Condition: | If the attribute 'Name of data element' is not unique within a Registration_Authority this attribute is mandatory. | N/A | If the attribute *name* is not unique within a *context*, this attribute is mandatory. |
| Data type: | Character | String | String |
| Comment: | Assignment of a unique identifier may be made mandatory as part of the registration procedure of any Registration_Authority. | | The requirement for an *item identifier* can be eliminated by qualifying *name* and/or *context name* to ensure that the combination is unique. |

##### D.2.1.6.1 Path from Administered_Item:

"Administered_Item". "administered item administration record". "Administration_Record". "administered item identifier".

#### D.2.1.7 Item registration authority identifier

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Registration_Authority | "Item Identifier". "item registration authority identifier" | item identifier – item registration authority identifier |
| Definition: | Any organization authorized to register data elements. | An identifier (described in ISO/IEC 11179 Part 6) assigned to the Registration_Authority registering the item. | An identifier (described in ISO/IEC 11179 Part 6) assigned to the registration authority registering the item. |
| Obligation: | Conditional | Mandatory | Conditional |
| Condition: | One Registration_Authority shall be specified for each Identifier present. | N/A | Required if *item identifier – data identifier* is not unique within the usage context. |
| Data type: | Character string | String | String |

##### D.2.1.7.1 Path from Administered_Item:

"Administered_Item". "administered item administration record". "Administration_Record". "administered item identifier".

#### D.2.1.8 Version

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Version | "Item identifier". "version" | Version |
| Definition: | Identification of an issue of a data element specification in a series of evolving data element specifications within a Registration_Authority. | The unique version identifier of the Administered_Item. | The unique version identifier of the metadata item. |
| Obligation: | Conditional | Mandatory | Optional |
| Condition: | This attribute is mandatory if updates on attributes occur which meet the maintenance rules for allocating new versions as set by the Registration | N/A | N/A |
| Data type: | Character | String | String |

##### D.2.1.8.1 Path from Administered_Item:

"Administered_Item". "administered item administration record". "Administration_Record". "administered item identifier".

### D.2.2 Common Definitional attributes

#### D.2.2.1 Definition

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Definition | "Definition (of Administered_Item)". "definition_text" | definition |
| Definition: | Statement that expresses the essential nature of a data element and permits its differentiation from all other data elements. | *Definition:* The definition of an Administered_Item within a Context.<br>*Definition text:* The text of the definition. | The definition of an metadata item within a context. |
| Obligation: | Mandatory | Mandatory | Mandatory |
| Data type: | Character string | String | String |
| Comment: | | Where more than one Definition is provided within a particular context, one of them may be specified as preferred by setting the attribute "preferred definition" to "True". | |

#### D.2.2.1.1 Path from Administered_Item:

"Administered_Item". "*administered item context*". "Terminological Entry". "*terminological entry languages*". "Language Section". "*definition entry*".

### D.2.2.2 Definition language

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Language Section". "language section language_identifier" | definition_language_identifier |
| Definition: | N/A | The identifier of the language used within the *Terminological Entry*, which applies to both the name and the definition. | The identifier of the language used within the definition. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | String | String |

#### D.2.2.2.1 Path from Administered_Item:

"Administered_Item". "*administered item context*". "Terminological Entry".

### D.2.2.3 Definition source reference

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Definition". "definition_source_reference" | definition_source_reference |
| Definition: | N/A | A reference to the source from which the definition is taken. | A reference to the source from which the definition is taken. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | String | String |

#### D.2.2.3.1 Path from Administered_Item:

"Administered_Item". "*administered item context*". "Terminological Entry". "*terminological entry languages*". "Language Section". "*definition entry*".

## D.2.3 Common Administrative attributes

### D.2.3.1 Comments

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Comments | "Administration_Record". "explanatory_comment" | Comments |
| Definition: | Remarks on the data element. | Descriptive comments about the Administered_Item. | Descriptive comments about the metadata item. |
| Obligation: | Optional | Optional | Optional |
| Data type: | Character string | String | String |

#### D.2.3.1.1 Path from Administered_Item:

"Administered_Item". "administered item administration record".

#### D.2.3.2 Registration status

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Registration status | “Administration_Record”. “registration_status” | registration_status |
| Definition: | A designation of the position in the registration life-cycle of a data element. | A designation of the status in the registration life-cycle of an Administered_Item. | A designation of the status in the registration life-cycle of a metadata item. |
| Obligation: | Conditional | Mandatory | Optional |
| Condition: | This attribute is mandatory during the data element life-cycle specified by any Registration_Authority. | N/A | N/A |
| Data type: | Character | String | String |
| Comment: | The type of registration_status to be distinguished and the allocation of the registration_status shall follow the rules that are described in the procedures for the registration of data elements (see Part 6 of this International Standard). | | |

##### D.2.3.2.1 Path from Administered_Item:

Administered_Item”. “administered item administration record”.

#### D.2.3.3 Responsible organization

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Responsible organization | "Organization" . "organization name" | Responsible organization |
| Definition: | The organization or unit within an organization that is responsible for the contents of the mandatory attributes by which the data element is specified. | *Organization:* A unique framework of authority, within which a person or persons act, or are designated to act, towards some purpose.<br>*stewardship:* The relationship of an Administered_Item, a Contact and an Organization involved in the stewardship of the metadata. | The organization or unit within an organization that is responsible for the contents of the mandatory attributes by which the metadata item is specified. |
| Obligation: | Optional | Mandatory | Optional |
| Data type: | Character string | String | String |
| Comment: | The organization shall be considered as 'owner' of the data element. | | |

##### D.2.3.3.1 Path from Administered_Item:

“Administered_Item” . “stewardship” .

#### D.2.3.4 Submitting organization

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Submitting organization | "Organization". "organization name" | Submitting organization |
| Definition: | The organization or unit within an organization that has submitted the data element for addition, change or cancellation/withdrawal in the data element dictionary. | *Organization:* A unique framework of authority, within which a person or persons act, or are designated to act, towards some purpose.<br>*submission:* The relationship of an Administered_Item, a Contact and an Organization involved in a submission of metadata. | The organization or unit within an organization that has submitted the metadata item for addition, change or cancellation/withdrawal in a metadata registry. |
| Obligation: | Optional | Mandatory | Optional |
| Data type: | Character string | String | String |

##### D.2.3.4.1 Path from Administered_Item:

"Administered_Item". "submission".

## D.2.4 Common Relational attributes

### D.2.4.1 Classification scheme name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Classification scheme | "Designation (of Administered_Item)". "name"<br>NOTE The "Administered_Item" referred to here is the Classification_Scheme, not the Administered_Item which is being classified. | Classification scheme name |
| Definition: | A reference to (a) class(es) of a scheme for the arrangement or division of objects into groups based on characteristics that the objects have in common, e.g. origin, composition, structure, application, function etc. | *Classification_Scheme:* The descriptive information for an arrangement or division of objects into groups based on characteristics which the objects have in common.<br>*name:* A name by which an Administered_Item (in this case the Classification_Scheme) is known within a specific Context. | The name of a particular arrangement or division of objects into groups based on characteristics which the objects have in common. |
| Obligation: | Optional | Conditional | Conditional |
| Condition: | N/A | If a Classification_Scheme is used, its name is mandatory. | If a Classification_Scheme is used, its name is mandatory. |
| Data type: | Character string | String | String |
| Comment | The definition does not specify whether the reference is by name or identifier. | | |

#### D.2.4.1.1 Path from Administered_Item:

"Administered_Item" (1). "*administered item classification*". "Classification_Scheme Item". "*classification scheme membership*". "Classification_Scheme". "classification scheme administration record". "Administration_Record". "Administered_Item" (2). "*administered item context*". "Terminological Entry". "*terminological entry languages*". "Language Section". "name entry".

NOTES (1) (2) The first "Administered_Item" is the one which is being classified. The second "Administered_Item" is the Classification_Scheme itself.

#### D.2.4.2 Classification scheme identifier

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Classification scheme | "Classification_Scheme". "classification scheme administration record". "Administration_Record". "administered item identifier" | classification scheme identifier |
| Definition: | A reference to (a) class(es) of a scheme for the arrangement or division of objects into groups based on characteristics that the objects have in common, e.g. origin, composition, structure, application, function etc. | *Classification_Scheme:* The descriptive information for an arrangement or division of objects into groups based on characteristics which the objects have in common.<br>*administered item identifier:* An identifier for an Administered_Item (in this case the Classification_Scheme) within a Registration_Authority. | The identifier of a particular arrangement or division of objects into groups based on characteristics which the objects have in common. |
| Obligation: | Optional | Conditional | Optional |
| Condition | N/A | If a Classification_Scheme is used, its administered item identifier is mandatory. | N/A |
| Data type: | Character string | String | String |
| Comment | The definition does not specify whether the reference is by name or identifier. | | |

##### D.2.4.2.1 Path from Administered_Item:

"Administered_Item" . "*administered item classification*" . "Classification_Scheme Item" . "*classification scheme membership*" .

#### D.2.4.3 Classification scheme type name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Classification_Scheme". "classification scheme type name" | Classification scheme type name |
| Definition: | N/A | The name of the type of classification scheme. | The name of the type of classification scheme. |
| Obligation: | N/A | Conditional | Optional |
| Condition | N/A | If Classification_Scheme is present, *classification scheme type name* is mandatory. | N/A |
| Data type: | N/A | String | String |

##### D.2.4.3.1 Path from Administered_Item:

"Administered_Item". "*administered item classification*". "Classification_Scheme Item". "*classification scheme membership*".

#### D.2.4.4 Classification scheme item type name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Classification_Scheme Item" . "classification scheme item type name | classification scheme item type name |
| Definition: | N/A | The name of the type of the classification scheme item. | The name of the type of the classification scheme item. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | String | String |

##### D.2.4.4.1 Path from Administered_Item:

"Administered_Item". "*administered item classification*".

#### D.2.4.5 Classification scheme item value

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Keyword | "Classification_Scheme Item". "classification scheme item value" | classification scheme item value |
| Definition: | One or more significant words used for retrieval of data elements. | An instance of a classification scheme item. | An instance of a classification scheme item. |
| Obligation: | Optional | Optional | Optional |
| Data type: | Character string | String | |
| Comment: | This attribute can be used for recording keywords (search keys) associated with the data element in question. | This edition of this part of ISO/IEC 11179 treats keywords as a type of classification scheme, with individual keywords being represented as classification scheme item values. | This edition of this part of ISO/IEC 11179 treats keywords as a type of classification scheme, with individual keywords being represented as classification scheme item values. |

##### D.2.4.5.1 Path from Administered_Item:

"Administered_Item". "administered item classification".

### D.2.4.6 Related metadata reference

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Related data reference | "Administration_Record". "administrative_note"<br>OR<br>"Administration_Record". "explanatory_comment"<br>OR<br>"Reference_Document". "reference_document_identifier" | Related metadata reference |
| Definition: | A reference between the data element and any related data. | *administrative_note:* any general note about the *Administered_Item*<br>*expanatory comment:* descriptive comments about the *Administered_Item*<br>*Reference_Document:* a document that provides pertinent details for consultation about a subject.<br>*reference_document_identifier:* An identifier for the Reference_Document. | A reference from one metadata item to another. |
| Obligation: | Optional | Optional | Optional |
| Data type: | Character string | String | String |
| Comment: | If this attribute occurs it shall be specified in pair with the attribute 'Type of relationship' | | |

#### D.2.4.6.1 Path from Administered_Item:

For "adminisftrative note":

Administered_Item". "administered item administration record".

For "explanatory_comment":

Administered_Item". "administered item administration record".

For "reference_document_identifier":

"Administered_Item". "reference"

#### D.2.4.7 Type of relationship

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Type of relationship | “Reference_Document”. “reference_document_type_description” | Type of relationship |
| Definition: | An expression that characterizes the relationship between the data element and related data. | The description of the type of association with another data element concept that this data element concept modifies, is modified by, or is otherwise linked with. | The description of the type of relationship identified by the related metadata reference. |
| Obligation: | Conditional | Conditional OR Optional | Conditional |
| Condition: | This attribute is mandatory if the attribute 'related data reference' occurs. | “reference_document_type_description” is optional if Reference_Document is used. | This attribute is mandatory if the attribute 'related metadata reference' occurs. |
| Data type: | Character string | String | String |
| Comment: | Examples of type of relationships are: 'qualifier of', 'qualified by', 'subject of', 'part of', 'physical condition', 'external reference', 'higher standard', 'data element concept'. | See C.2.4.6 Related metadata reference. | |

##### D.2.4.7.1 Path from Administered_Item:

“Administered_Item”. “reference”

### D.2.5 Attributes specific to Data_Element_Concepts

#### D.2.5.1 Object class name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | “Designation (of Administered)”. “name”<br>NOTE The Administered item referred to here is the Object_Class. | Object class name |
| Definition: | N/A | *data element concept object class:* the designation of an Object_Class for a Data_Element_Concept.<br>*name:* A name by which an Administered_Item (in this case the Object_Class) is known within a specific context. | The designation of an *object class* for a *data element concept*. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | String | String |

#### D.2.5.1.1 Path from Data_Element_Concept:

“Data_Element_Concept”. “data element concept object class”. “Object_Class”. “object class administration record”. “Administration_Record”. “Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “name entry”.

### D.2.5.2 Object class identifier

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | “Administration_Record”. “administered item identifier”<br>NOTE The Administered item referred to here is the Object_Class. | Object class identifier |
| Definition: | N/A | *administered item identifier:* An identifier for an Administered_Item (in this case the Object_Class) within a Registration_Authority. | The identifier of an *object class* for a *data element concept*. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | String | String |

#### D.2.5.2.1 Path from Data_Element_Concept:

“Data_Element_Concept”. “data element concept object class”. “Object_Class”. “object class administration record”.

### D.2.5.3 Property name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | “Designation (of Administered)”. “name”<br>NOTE The Administered item referred to here is the Property. | Property name |
| Definition: | N/A | *data element concept property:* the designation of a Property for a Data_Element_Concept.<br>*name:* A name by which an Administered_Item (in this case the Property) is known within a specific context. | The designation of a *property* for a *data element concept*. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | String | String |

#### D.2.5.3.1 Path from Data_Element_Concept:

“Data_Element_Concept”. “data element concept property”. "Property”. “property administration record". “Administration_Record”. “Administered_Item”. “*administered item context*”. “Terminological Entry”. “*terminological entry languages*”. “Language Section”. “name entry”.

#### D.2.5.4 Property identifier

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Administration_Record". "administered item identifier"<br>NOTE The Administered item referred to here is the Property. | Property identifier |
| Definition: | N/A | *administered item identifier:* An identifier for an Administered_Item (in this case the Property) within a Registration_Authority. | The identifier of a *property* for a *data element concept*. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | String | String |

##### D.2.5.4.1 Path from Data_Element_Concept:

"Data_Element_Concept". "data element concept property". "Property". "property administration record".

### D.2.6 Attributes specific to Data_Elements

#### D.2.6.1 Representation category

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Representation category | Not supported. | Not supported. |
| Definition: | Type of symbol, character or other designation used to represent a data element. | N/A | N/A |
| Obligation: | Mandatory | N/A | N/A |
| Data type: | Character string | N/A | N/A |
| Comment: | The representation category shall be specified by the relevant standard.<br>Examples of possible representation categories:<br>— character representation (ISO/IEC 646)<br>— character/symbol representation (ISO registration no. 143)<br>— bar coded representation (EIA-556)<br>— graphical representation | | |

### D.2.6.2 Representation class

EDITOR'S NOTE #53. Issue 114 proposes to remove *Representation Class*. How should we map 'form of representation' in Edition 3?

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Form of representation | "Designation (of Administered)". "name"<br>NOTE The Administered item referred to here is the Representation Class. | Representation class |
| Definition: | Name or description of the form of representation for the data element, e.g. 'quantitative value', 'code', 'text', 'icon'. | *Representation Class:* the classification of types of representations.<br>*name:* A name by which an Administered_Item (in this case the Representation Class) is known within a specific context. | The name of the class of representation of a data element. |
| Obligation: | Mandatory | Optional | Optional |
| Data type: | Character string | String | String |
| Comment: | 1. See ISO/IEC 11179-2 for appropriate terms ('property words' or 'class words') to be used.<br>2. Example 1: For the data element named: 'country of origin code' this attribute contains: 'code'.<br>3. Example 2: For the data element: 'product description' this attribute contains: 'text'.<br>4. Example 3: For the data element: 'weight of consignment' this attribute contains: 'quantitative value'. | See 4.13.1.4 for a list of Representation Class terms. | See 4.13.1.4 for a list of Representation Class terms. |

#### D.2.6.2.1 Path from Data_Element

"Data_Element". "*data element representation class*". "Representation Class". "Administration_Record". "Administered_Item". "*administered item context*". "Terminological Entry". "*terminological entry language*s". "Language Section". "name entry".

#### D.2.6.3 Value domain name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not directly supported. | "Designation (of Administered)". "name"<br>NOTE The Administered item referred to here is the Value_Domain. | value domain name |
| Definition: | N/A | *Value domain:* A set of permissible values. It provides representation, but has no implication as to what data element concept the values may be associated with nor what the values mean.<br>*name:* A name by which an Administered_Item (in this case the Value_Domain) is known within a specific context. | The name of the value domain that provides representation for the data element. |
| Obligation: | N/A | Mandatory | Optional |
| Data type: | N/A | String | String |
| Comment: | The closest equivalent is "permissible data element values" (see F.2.6.10), but this actually represents the values. | | |

##### D.2.6.3.1 Path from Data_Element

"Data_Element". "*data element representation*". "Value_Domain". "value domain administration record". "Administration_Record". "Administered_Item". "*administered item context*". "Terminological Entry". "*terminological entry languages*". "Language Section". "name entry".

#### D.2.6.4 Value domain identifier

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not directly supported. | "Administration_Record". "administered item identifier" | value domain identifier |
| Definition: | N/A | *Value_Domain:* A set of permissible values. It provides representation, but has no implication as to what Data_Element_Concept the values may be associated with nor what the values mean.<br>*administered item identifier:* An identifier for an administered item (in this case the Value_Domain) within a registration authority. | The identifier of the value domain that provides representation for the data element. |
| Obligation: | N/A | Mandatory | Optional |
| Data type: | N/A | String | String |

| | | | |
|---|---|---|---|
| Comment: | The closest equivalent is “permissible data element values” (see F.2.6.10), but this actually represents the values. | | |

#### D.2.6.4.1 Path from Data_Element

“Data_Element”. “*data element representation*”. “Value_Domain”. “value domain administration record”.

### D.2.6.5 Datatype name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Datatype of data element values | “Datatype” . “datatype_name” | datatype_name |
| Definition: | A set of distinct values for representing the data element value. | *Datatype:* A set of distinct values characterized by properties of those values and by operations on those values.<br>*datatype_name:* A designation for the datatype. | *datatype_name:* A designation for the datatype. |
| Obligation: | Mandatory | Mandatory | Conditional |
| Condition | N/A | N/A | Required if neither *value domain name* nor *value domain identifier* is specified. |
| Data type: | Character string | String | String |
| Comment: | Examples: Possible instances are: 'character', 'ordinal number', 'integer', 'real', 'scaled', 'bit', 'rational'.<br>Note: The examples suggest the attribute is intended to be the name of the datatype, whereas the definition implies it is a set of values. | In the metamodel, the datatype is an attribute of the value domain, not directly of the data element. | |

#### D.2.6.5.1 Path from Data_Element

“Data_Element”. “*data element representation*”. “Value_Domain”. “value_domain_datatype”.

### D.2.6.6 Datatype scheme reference

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | “Datatype”. “datatype_scheme_reference” | Datatype scheme reference |
| Definition: | N/A | A reference identifying the source of the Datatype specification. | A reference identifying the source of the datatype specification. |
| Obligation: | N/A | Mandatory | Conditional |
| Condition | N/A | N/A | Required if *datatype_name* is specified. |
| Data type: | N/A | String | String |

| | | | |
|---|---|---|---|
| Comment: | | In the metamodel, the datatype is an attribute of the value domain, not directly of the data element. | |

#### D.2.6.6.1 Path from Data_Element

“Data_Element”. “*data element representation*”. “Value_Domain”. “value_domain_datatype”.

### D.2.6.7 Maximum size

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Maximum size of data element values | “Value_Domain”. “value_domain_maximum_character_quantity” | Maximum size |
| Definition: | The maximum number of storage units (of the corresponding datatype) to represent the data element value. | The maximum number of characters to represent the data element value.<br>NOTE Applicable only to character Datatypes. | The maximum number of storage units (of the corresponding datatype) to represent the data element value. |
| Obligation: | Mandatory | Optional | Optional |
| Data type: | Integer | Integer | Integer |
| Comment: | 1. Example 1:<br>For data element: 'invoice number' the attribute 'datatype' has instance 'character'<br>and the attribute 'maximum size of data element value' has value: '17'. The data<br>element value of 'invoice number' shall have a maximum of 17 characters.<br>2. The two attributes 'maximum and minimum (see 6.4.5) size of data element<br>values' indicate whether data element values are 'fixed' (maximum and minimum<br>size are equal) or 'variable' (maximum and minimum size vary). | This is not exactly equivalent, because it applies only to character datatypes. | |

#### D.2.6.7.1 Path from Data_Element

“Data element”. “data element representation”.

### D.2.6.8 Minimum size

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Minimum size of data element values. | Not supported. | Minimum size |
| Definition: | The minimum number of storage units (of the corresponding datatype) to represent the data element value. | N/A | The minimum number of storage units (of the corresponding datatype) to represent the data element value. |
| Obligation: | Mandatory | N/A | Optional |
| Data type: | Integer | N/A | Integer |
| Comment: | 1. Example 1:<br>For data element: 'product description' the attribute 'datatype' has instance 'character' and the attribute 'minimum size of data element value' has instance: '10'.<br>The data element value of 'product description' shall have a minimum of 10 characters.<br>2. The two attributes 'maximum (see 6.4.4) and minimum size of data element values' indicate whether data element values are 'fixed' (maximum and minimum size are equal) or 'variable' (maximum and minimum size vary). | | |

### D.2.6.9 Layout of representation

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Layout of representation | Not supported. | Layout of representation |
| Definition: | The layout of characters in data element values expressed by a character string representation. | N/A | The layout of characters in data element values expressed by a character string representation. |
| Obligation: | Conditional | N/A | Optional |
| Condition: | If the data element is of the class 'quantitative data' this attribute is mandatory. If the attribute 'form of representation' is 'code' the use of this attribute is recommended if the code representation has to have a specific structure or layout. | | |
| Data type: | Character string | | String |
| Comment: | 1. For quantitative data it is necessary to distinguish between integers, decimal mark and floating point notations.<br>Example:<br>Integers may be indicated with 'n', for decimal mark the number of characters before and after the decimal mark are specified as: n(5).n(3), for floating point notations the layout convention for a value with exponents shall comply with ISO 6093: n(3).n(3)E2, where 'E2' stands for max. 2 digits for the power of 10.<br>2. For code representations having a specific structure or layout the type of character for each position in the code structure is important for validation purposes.<br>Example:<br>The data element 'flight number' has an international code representation structure consisting of two alphabetic characters of the airline company followed by a three-digit number identifying the flight (from starting-point to destination).<br>The contents of the attribute: 'layout of representation' is: 'AA999'. | | |

#### D.2.6.10 Permissible data element values

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Permissible data element values | See Value_Domain for equivalent capability. | See Value_Domain for equivalent capability. |
| Definition: | The set of representations of permissible instances of the data element, according to the representation form, layout, datatype and maximum and minimum size specified in the corresponding attributes. The set can be specified by name, by reference to a source, by enumeration of the representation of the instances or by rules for generating the instances. | N/A | N/A |
| Obligation: | Mandatory | N/A | N/A |
| Data type: | Character string | N/A | N/A |
| Comment: | When the permissible data element values are an enumeration of coded representations each data element value and instance shall be presented as a pair. | | |

### D.2.7 Attributes specific to Conceptual_Domains

#### D.2.7.1 Dimensionality

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Conceptual_Domain" . "dimensionality" | dimensionality |
| Definition: | N/A | The dimensionality for a concept. | The dimensionality for a concept. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | String | String |
| Comment: | | For example, length, mass, velocity, currency. | For example, length, mass, velocity, currency. |

## D.2.8 Attributes specific to Value_Domains

### D.2.8.1 Datatype name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | See “Datatype of data element values” (F.2.6.5) | “Value_Domain”. “value_domain_datatype”. “Datatype”. “datatype_name” | datatype_name |
| Definition: | N/A | *Datatype:* A set of distinct values characterized by properties of those values and by operations on those values.<br>*datatype_name:* A designation for the datatype. | *datatype_name:* A designation for the datatype. |
| Obligation: | | Mandatory | Mandatory |
| Data type: | | String | String |

### D.2.8.2 Datatype scheme reference

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | “Value_Domain”. “value_domain_datatype”. “Datatype”. “datatype_scheme_reference” | Datatype scheme reference |
| Definition: | | A reference identifying the source of the datatype specification. | A reference identifying the source of the datatype specification. |
| Obligation: | | Mandatory | Optional |
| Data type: | | String | String |

### D.2.8.3 Unit of measure name

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | “Value_Domain”. “value_domain_unit_of_measure”. “Unit_of_Measure”. “unit of measure name” | unit of measure name |
| Definition: | | The name of a unit of measure. | The name of a unit of measure. |
| Obligation: | | Optional | Optional |
| Data type: | | String | String |

## D.2.9 Attributes specific to Permissible_Values

### D.2.9.1 Value

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | See “permissible data element values” (F.2.6.10) | “Permissible_Value”. “permitted_value”. “Value”. “value item” | value |
| Definition: | N/A | A representation of a value meaning in a specific value domain. The actual value. | A representation of a value meaning in a specific value domain. The actual value. |
| Obligation: | N/A | Mandatory | Mandatory |
| Data type: | N/A | String | String |
| Comment: | | | |

### D.2.9.2 Permissible_Value Begin Date

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | “Permissible_Value”. “permissible_value_begin_date” | permissible_value_begin_date |
| Definition: | N/A | The date this value became/becomes permissible in the value domain. | The date this value became/becomes permissible in the value domain. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | Date | Date |

### D.2.9.3 Permissible_Value End Date

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | “Permissible_Value”. “permissible_value_end_date” | permissible_value_end_date |
| Definition: | N/A | The date this value became/becomes no longer permissible in the value domain. | The date this value became/becomes no longer permissible in the value domain. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | Date | Date |

## D.2.10 Attributes specific to Value_Meanings

### D.2.10.1 Value meaning description

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Value_Meaning". "value meaning description" | value meaning description |
| Definition: | N/A | A description of a value meaning. | A description of a value meaning. |
| Obligation: | N/A | Mandatory | Mandatory |
| Data type: | N/A | String | String |

### D.2.10.2 Value meaning identifier

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Value_Meaning". "value meaning identifier" | value meaning identifier |
| Definition: | N/A | The unique identifier for a value meaning. | The unique identifier for a value meaning. |
| Obligation: | N/A | Mandatory | Optional |
| Data type: | N/A | String | String |

### D.2.10.3 Value meaning begin date

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Value_Meaning". "value_meaning_begin_date" | value_meaning_begin_date |
| Definition: | N/A | The effective_date of this value meaning in the conceptual domain. | The effective_date of this value meaning in the conceptual domain. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | Date | Date |

### D.2.10.4 Value meaning end date

| | 1994 Clause 6 | 2002 Clause 4 | 2002 Clause 5 |
|---|---|---|---|
| Attribute name: | Not supported. | "Value_Meaning". "value_meaning_end_date" | value_meaning_end_date |
| Definition: | N/A | The date this value meaning became/becomes invalid. | The date this value meaning became/becomes invalid. |
| Obligation: | N/A | Optional | Optional |
| Data type: | N/A | Date | Date |

# Annex E
(informative)

# Mapping the ISO/IEC 11179-3:2002 metamodel to the ISO/IEC 11179-3:200n metamodel

EDITOR'S NOTE #54. Mapping from Edition 2 to Edition 3 to be added.

## E.1 Introduction

This Annex explains how the Edition 2 metamodel relates to the Edition 3 metamodel.

## E.2 Mapping the Edition 2 Common Facilities

To be added.

| Edn. 2 clause # | Edition 2 metamodel object | Edn. 3 clause # | Edition 3 metamodel object |
|---|---|---|---|
| | | | |

## E.3 Mapping the Data Description Model

To be completed.

| Edn. 2 clause # | Edition 2 metamodel object | Edn. 3 clause # | Edition 3 metamodel object |
|---|---|---|---|
| | | | |
| | Conceptual_Domain Relationship | | 'Relation' in the new 'Concept_System' metamodel region |
| | conceptual domain relationship type description | | As a Registered_Item, 'Relation' can be designated and defined using the common facilities of the metamodel. |
| | conceptual domain representation | | value_domain_meaning |
| | contact name | | contact person |
| | context description | | Definition for Context. |
| | context description language_identifier | | Language of Definition of Context |
| | country identifier | | region_identifier |
| | | | |

# Annex F
(informative)

# Concept System Examples

This Annex illustrates the use of the Concept System region (see 8.1).

## F.1 Concept System Metamodels

The concept system metamodel specified in 8.1 is very generic, so that it may be used to register concept systems that are defined in a wide range of formalisms. Most formalisms will have some built-in constructs which are not built-in to the 11179-3 metamodel. The concept system metamodel is generic enough to also support registration of such built-in constructs, in a "notation" concept system. For example, an OWL concept system can be used to define OWL ontological relations such as rdf:type, rdfs:range, and owl:disjointWith.

By registering a full metamodel of each such formalism also as a concept system, the same facility can also be used to describe mappings between them, as well as to the 11179-3 concept system metamodel itself. The table below summarizes some suggested primary mappings between 11179-3 and a selection of notations. Note that in the RDF-based notations (SKOS and OWL) it is suggested to treat inverse properties as roles of an implicit binary relation. Some further details are called out below.

| Notation | Concept | Relation | Relation_Role | Link |
|---|---|---|---|---|
| SKOS | Concept | — | *semantic relations* | Statement |
| OWL | Class *or* Thing | — | ObjectProperty | Statement |
| UML | Class *or* Object | Association | AssociationEnd | Link |
| ORM | non-lexical object *or* non-lexical object type | idea type | role | idea |
| XTM | Topic | Association Type | Association Role | Association |
| SBVR | concept *or* characteristic | (non-unary) fact type | fact type role | fact |

**Table F-1: Correspondences of 11179-3 concept system metamodel to selected notations**

SKOS

There is no metaclass in SKOS for semantic relations, instead there are only the foundational broader, narrower and related properties, and a semantic relation is one defined by those properties or by subproperties of those properties.

OWL

Some OWL built-in constructs are most naturally described as ternary relations, and others as variable arity relations with two roles. It is also reasonable to describe object properties as relations rather than relation roles. See OWL Example below.

UML

Because the 11179-3 metamodel does not impose a meta-level hierarchy, the Generalization metaclass in UML can be interpreted as a built-in relation, and generalization relationships thus as links (in the 11179-3 sense) between UML Classes.

ORM

There is no official standard for ORM, but in this appendix the ORM metamodel provided in ISO TR 9007 (1987) is taken as normative.

XTM

An XML syntax for Topic Maps (ISO/IEC 13250).

SBVR

It is most natural to treat SBVR characteristics (unary fact types) as concepts in 11179-3, rather than relations, because links in 11179-3 are required to have at least two ends. See SBVR Example below.

## F.2 SKOS Example

This is a very simple example using SKOS (Simple Knowledge Organization System)[4].

### F.2.1 SKOS Metamodel

The core of the SKOS (meta)model[5] contains only two semantic relations[6], defined by three RDF properties. Describing these two semantic relations in terms of the draft 11179-3 metamodel is very straightforward:

| <Concept_System> | | | |
|---|---|---|---|
| | **notation** | **referencedConceptSystem** | **importedConceptSystem** |
| SKOS-CORE | | | |

**Table F-2: SKOS-CORE as a 11179 Concept System**

| <Binary_Relation> | | | | | |
|---|---|---|---|---|---|
| | **container** | **role** | **reflexivity** | **symmetry** | **transitivity** |
| generalization | SKOS-CORE | skos:broader | | antisymmetric | transitive |
| | | skos:narrower | | | |
| association | SKOS-CORE | skos:related | | symmetric | intransitive |

**Table F-3: SKOS relations as a 11179 Binary Relations**

### F.2.2 SKOS Example Thesaurus

Our SKOS example is a simple thesaurus of marital statuses. Here is its expression in Turtle:

---

[4] See http://www.w3.org/TR/skos-primer/

[5] See http://www.w3.org/TR/skos-primer/#secsimple

[6] See http://www.w3.org/TR/skos-primer/#secrel

```
:MaritalStatus rdf:type skos:ConceptScheme .

:Married rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:related :Single .

:Single rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:narrower :NeverMarried .

:NeverMarried rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;

:Widowed rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:broader :Single .

:Divorced rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:broader :Single .
```

Here is a description of the above thesaurus in terms of the draft 11179-3 metamodel:

| <Concept_System> | | | |
|---|---|---|---|
| | **notation** | **referencedConceptSystem** | **importedConceptSystem** |
| MaritalStatus | SKOS/Turtle | SKOS-CORE | |

**Table F-4: SKOS Thesaurus Example – 11179 Concept System**

| <Concept> | |
|---|---|
| | **container** |
| ms:Married | MaritalStatus |
| ms:Single | MaritalStatus |
| ms:NeverMarried | MaritalStatus |
| ms:Widowed | MaritalStatus |
| ms:Divorced | MaritalStatus |

**Table F-5: SKOS Thesaurus Example – 11179 Concepts**

| <Link> | | | |
|---|---|---|---|
| **container** | **relation** | **link_end** | |
| | | **end_role** | **end** |
| MaritalStatus | association | skos:related | ms:Married |
| | | skos:related | ms:Single |
| MaritalStatus | generalization | skos:broader | ms:Single |
| | | skos:narrower | ms:NeverMarried |

| | | | |
|---|---|---|---|
| MaritalStatus | generalization | skos:broader | ms:Single |
| | | skos:narrower | ms:Widowed |
| MaritalStatus | generalization | skos:broader | ms:Single |
| | | skos:narrower | ms:Divorced |

**Table F-6: SKOS Thesaurus Example – 11179 Links**

### F.2.3 Example Value Domain References

To illustrate one possible connection to data description, here is an example pair of enumerated value domains described with references to the above SKOS thesaurus. (Note that unused attributes have been omitted from these descriptions.)

| <Enumerated_Conceptual_Domain> | |
|---|---|
| | **member** |
| binary_ms_CD | ms:Single |
| | ms:Married |
| specific_ms_CD | ms:NeverMarried |
| | ms:Married |
| | ms:Widowed |
| | ms:Divorced |

**Table F-7: SKOS Thesaurus Example – 11179 Conceptual Domains**

| <Enumerated_Value_Domain> | | | | |
|---|---|---|---|---|
| | **datatype** | **meaning** | **member** | |
| | | | **value** | **meaning** |
| binary_marital_status | Bit | binary_ms_CD | 0 | ms:Single |
| | | | 1 | ms:Married |
| marital_status_code | Character | specific_ms_CD | 'S' | ms:NeverMarried |
| | | | 'M' | ms:Married |
| | | | 'W' | ms:Widowed |
| | | | 'D' | ms:Divorced |

**Table F-8: SKOS Thesaurus Example – 11179 Value Domains**

## F.3 ORM Example

This example uses the Object Role Modelling[7] (ORM) model from ISO TR 9007:1987 Appendix E.

### F.3.1 ORM Metamodel

There are two relations built-into ORM which need to be registered first. They are described in TR 9007 using the PASCAL syntax defined in TR 9007 Appendix C. The subtype relation is defined by the declaration:

```
(R5) nolot-subtype = "NOLOT called"
                           (nolot-name-1 | nolot-name-1-list)
                     "is subtype-of NOLOT called" nolot-name-2 ";".
```

The other relation is implicitly defined within the idea-declaration definition:

```
(R7) idea-declaration = "IDEA (with-first ROLE (called" role-name-1
                                "and on NOLOT called" nolot-name-1 ")"
                        "and with-second ROLE (called" role-name-2
                                "and on NOLOT called" nolot-name-2 "))"
                        "is called" idea-name ";".
```

For clarity, we will first rewrite the idea-declaration rule as follows:

```
(R7a) idea-declaration = "IDEA (with-first" idea-role-1
                          "and with-second" idea-role-2 ")"
                          "is called " idea-name ";".
(R7b) idea-role        = "ROLE (called" role-name
                          "and on NOLOT called" nolot-name ")".
(R7c) idea-role-1      = idea-role.
(R7d) idea-role-2      = idea-role.
```

Rule (R7a) corresponds to relation_role_set in the 11179-3 metamodel; rule (R7b) defines a relation which is not built-in to the 11179-3 metamodel. The ORM relations defined by rules (R5) and (R7b) thus may be described in terms of the 11179-3 metamodel as follows:

| <Concept_System> | | | |
|---|---|---|---|
| | **notation** | **referencedConceptSystem** | **importedConceptSystem** |
| ORM | PASCAL | | |

**Table F-9: ORM as a 11179 Concept System**

| <Binary_Relation> | | | | | |
|---|---|---|---|---|---|
| | **container** | **role** | **reflexivity** | **symmetry** | **transitivity** |
| subtype | ORM | subtype-of | | antisymmetric | transitive |
| | | supertype-of | | | |
| role-on | ORM | on | | | |
| | | role | | | |

**Table F-10: ORM Relations as 11179 Binary Relations**

[7] see http://www.orm.net/

| <Relation_Role> | | |
|---|---|---|
| | **multiplicity** | **ordinal** |
| subtype-of | | 1 |
| supertype-of | | 2 |
| on | 1 | 1 |
| role | | 2 |

**Table F-11: ORM Roles as 11179 Relation Roles**

### F.3.2 Car Registration Model

The example model is depicted graphically in Figure E18, reproduced below:

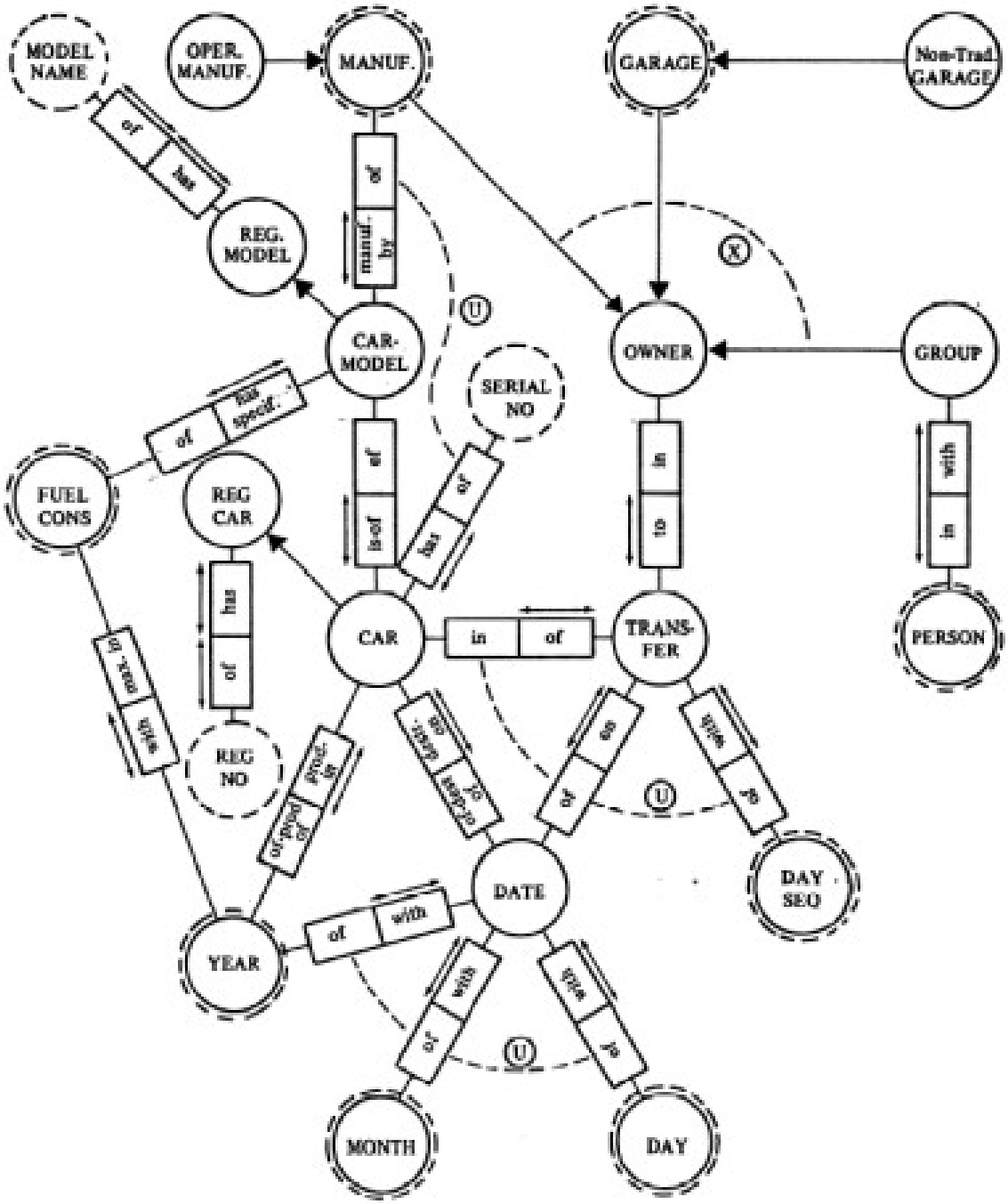



**Figure F-1 — Car Registration Model in ORM**

The textual definition is reproduced below:

```
begin
add CONCEPTUAL-SCHEMA called 'CAR-REGISTRATION' ;
add NOLOT called ('MANUFACTURER' 'OPERATING-MANUFACTURER' 'REG-CAR'
```

```
                  'CAR' 'REG-MODEL' 'CAR-MODEL' 'FUEL-CONSUMPTION'
                  'DATE' 'YEAR' 'MONTH' 'DAY' 'TRANSFER' 'DAY-SEQ'
                  'OWNER' 'GARAGE' 'NON-TRADING-GARAGE' 'GROUP'
                  'PERSON');
add LOT called {'MANUFACTURER-NAME' 'REG-NO' 'SERIAL-NO' 'MODEL-NAME'
                'FUEL-CONSUMPTION-AMOUNT' 'YEAR-NO' 'MONTH-NO'
                'DAY-NO' 'SEQ-NO' 'GARAGE-NAME' 'PERSON-NAME');
add NOLOT called 'OPERATING-MANUFACTURER'
          is subtype-of NOLOT called 'MANUFACTURER';
add NOLOT called ('MANUFACTURER' 'GARAGE' 'GROUP')
          is subtype-of NOLOT called 'OWNER';

    Note: three other subtype declarations omitted here.

add IDEA (with-first ROLE (called 'manuf-by'
                           and on NOLOT called 'CAR-MODEL')
          and with-second ROLE (called 'of'
                           and on NOLOT called 'MANUFACTURER'))
          is called 'builds';

    Note: thirteen other idea declarations omitted here.

add BRIDGE (with-first ROLE (called 'called'
                             and on NOLOT called 'REG-CAR')
            and with-second ROLE (called 'of'
                             and on LOT called 'REG-NO'))
            is called 'registration';

    Note: two other explicit bridge declarations omitted here.

add BRIDGE (with-first ROLE (called 'called'
                             and on NOLOT called 'MANUFACTURER')
            and with-second ROLE (called 'of'
                             and on LOT called 'MANUFACTURER-NAME'))
            is called 'naming-of-model';

    Note: seven other implicit bridge declarations omitted here.
    Note: the list of constraints is given on the next pages

end.
```

It has been chosen to regard only the non-lexical object types as concepts, and thus only the subtype declarations as links, and the idea types as relations. For the omitted subtype declarations, we will assume:

```
add NOLOT called 'NON-TRADING-GARAGE'
          is subtype-of NOLOT called 'GARAGE';
add NOLOT called 'CAR-MODEL'
          is subtype-of NOLOT called 'REG-MODEL';
add NOLOT called 'CAR'
          is subtype-of NOLOT called 'REG-CAR';
```

For the omitted idea type declarations, we will not go through all of them, but will assume:

```
add IDEA (with-first ROLE (called 'is-of'
                           and on NOLOT called 'CAR-MODEL')
          and with-second ROLE (called 'of'
                           and on NOLOT called 'CAR'))
          is called 'model';

add IDEA (with-first ROLE (called 'to'
                           and on NOLOT called 'OWNER')
```

```
          and with-second ROLE (called 'in'
                              and on NOLOT called 'TRANSFER'))
          is called 'transfer-owner';

add IDEA (with-first ROLE (called 'of'
                              and on NOLOT called 'CAR')
          and with-second ROLE (called 'in'
                              and on NOLOT called 'TRANSFER'))
          is called 'transfer-car';

add IDEA (with-first ROLE (called 'in'
                              and on NOLOT called 'GROUP')
          and with-second ROLE (called 'with'
                              and on NOLOT called 'PERSON'))
          is called 'group-member';
```

The following description in terms of the 11179-3 Concept System metamodel results:

| <Concept_System> | | | |
|---|---|---|---|
| | **notation** | **referencedConceptSystem** | **importedConceptSystem** |
| CAR-REGISTRATION | ISO9007-E3 | ORM | |

**Table F-12: Car Registration Model - 11179 Concept System**

| <Concept> | |
|---|---|
| | **container** |
| MANUFACTURER | CAR-REGISTRATION |
| OPERATING-MANUFACTURER | CAR-REGISTRATION |
| REG-CAR | CAR-REGISTRATION |
| CAR | CAR-REGISTRATION |
| REG-MODEL | CAR-REGISTRATION |
| CAR-MODEL | CAR-REGISTRATION |
| FUEL-CONSUMPTION | CAR-REGISTRATION |
| DATE | CAR-REGISTRATION |
| YEAR | CAR-REGISTRATION |
| MONTH | CAR-REGISTRATION |
| DAY | CAR-REGISTRATION |
| TRANSFER | CAR-REGISTRATION |
| DAY-SEQ | CAR-REGISTRATION |
| OWNER | CAR-REGISTRATION |
| GARAGE | CAR-REGISTRATION |
| NON-TRADING-GARAGE | CAR-REGISTRATION |
| GROUP | CAR-REGISTRATION |
| PERSON | CAR-REGISTRATION |

**Table F-13: Car Registration Model - 11179 Concepts**

| <Binary_Relation> | | | | | |
|---|---|---|---|---|---|
| | **container** | **role** | **reflexivity** | **symmetry** | **transitivity** |
| builds | CAR-REGISTRATION | manuf-by | | | |
| | | of | | | |
| model | CAR-REGISTRATION | is-of | | | |
| | | of | | | |
| transfer-owner | CAR-REGISTRATION | to | | | |
| | | in | | | |
| transfer-car | CAR-REGISTRATION | of | | | |
| | | in | | | |
| group-member | CAR-REGISTRATION | in | | | |
| | | with | | | |

**Table F-14: Car Registration Model - 11179 Binary Relations**

| <Link> | | | |
|---|---|---|---|
| **container** | **relation** | **link_end** | |
| | | **end_role** | **end** |
| CAR-REGISTRATION | subtype | supertype-of | OPERATING_MANUFACTURER |
| | | subtype-of | MANUFACTURER |
| CAR-REGISTRATION | subtype | supertype-of | MANUFACTURER |
| | | subtype-of | OWNER |
| CAR-REGISTRATION | subtype | supertype-of | GARAGE |
| | | subtype-of | OWNER |
| CAR-REGISTRATION | subtype | supertype-of | GROUP |
| | | subtype-of | OWNER |
| CAR-REGISTRATION | subtype | supertype-of | NON-TRADING-GARAGE |
| | | subtype-of | GARAGE |
| CAR-REGISTRATION | subtype | supertype-of | REG-CAR |
| | | subtype-of | CAR |
| CAR-REGISTRATION | subtype | supertype-of | REG-MODEL |
| | | subtype-of | CAR-MODEL |
| CAR-REGISTRATION | role-on | role | builds.manuf-by |
| | | on | MANUFACTURER |

| | | | |
|---|---|---|---|
| CAR-REGISTRATION | role-on | role | builds.of |
| | | on | CAR-MODEL |
| CAR-REGISTRATION | role-on | role | model.is-of |
| | | on | CAR-MODEL |
| CAR-REGISTRATION | role-on | role | model.of |
| | | on | CAR |
| CAR-REGISTRATION | role-on | role | transfer-owner.to |
| | | on | OWNER |
| CAR-REGISTRATION | role-on | role | transfer-owner.in |
| | | on | TRANSFER |
| CAR-REGISTRATION | role-on | role | transfer-car.of |
| | | on | CAR |
| CAR-REGISTRATION | role-on | role | transfer-car.in |
| | | on | TRANSFER |
| CAR-REGISTRATION | role-on | role | group-member.in |
| | | on | GROUP |
| CAR-REGISTRATION | role-on | role | group-member.of |
| | | on | PERSON |

**Table F-15: Car Registration Model - 11179 Links**

## F.4 OWL Example

Because OWL (Web Ontology Language[8]) is founded upon the very simple binary predicate model of RDF, there is more than one reasonable way to map OWL into the 11179-3 Concept System metamodel. Perhaps the more obvious is to treat each ObjectProperty as a relation, each with relation roles rdf:subject and rdf:object. However, an analogy to Properties in UML and MOF will instead suggest treating each ObjectProperty as representing a relation role, of an underlying binary relation taken to be implicit in the OWL representation. Either approach is workable, and indeed one might even mix the two approaches, treating some ObjectProperties as relations and others as relation roles, based on some case-by-case evaluation of the relative merits of each treatment.

### F.4.1 OWL Metamodel

A convenient synopsis of OWL built-in constructs is provided in the OWL Web Ontology Language Overview. Some of the OWL built-in constructs correspond directly to elements of the 11179-3 Concept System metamodel. These are:

[8] see http://www.w3.org/TR/owl-features/

| OWL Constructs | 11179-3 metamodel description type |
|---|---|
| Ontology | Concept_System |
| imports | Importation |
| minCardinality, maxCardinality, cardinality, FunctionalProperty, InverseFunctionalProperty | multiplicity |
| TransitiveProperty | transitivity |
| OWL constructs with directly corresponding 11179-3 metamodel elements | |

We also may capture OWL *inverseOf* assertions using only built-in 11179-3 metamodel constructs, as will be shown below. (We thus may also describe all *domain* assertions as *range* assertions on the opposing role, and therefore omit *domain* from our OWL metamodel as well.) And since assertions of membership in *SymmetricProperty* can also be regarded as simply an alternate syntax for expressing reflexive *inverseOf* assertions, they too may be captured in the same way.

Many other OWL built-in constructs do not have corresponding elements built-in to the 11179-3 Concept System metamodel, as summarized in the following table:

| Group of OWL Constructs | 11179-3 metamodel description type |
|---|---|
| metaclasses Class, Property, ObjectProperty and DatatypeProperty | Concept |
| classes Thing and Nothing | Concept |
| datatypes | Concept |
| equivalentClass, equivalentProperty, sameAs | Binary_Relation (symmetric, transitive) |
| differentFrom, complementOf, disjointWith | Binary_Relation (symmetric, intransitive) |
| subClassOf, subPropertyOf | Binary_Relation (asymmetric, transitive) |
| type, range | Binary_Relation (asymmetric, intransitive) |
| AllDifferent (and distinctMembers) | Relation (variable arity, 1 role) |
| intersectionOf, oneOf, unionOf | Relation (variable arity, 2 roles) |
| allValuesFrom, someValuesFrom, hasValue | Relation (arity=3, 3 roles) |
| OWL built-in constructs described in OWL metamodel | |

Some of these constructs are actually reused from RDFS, which in turn is defined on top of RDF, and most of the OWL datatypes are taken from XML Schema. To describe this explicitly, we actually describe four interrelated metamodels to support OWL: one each for RDF, RDFS, and the subset of XML Schema used in OWL; and one for OWL proper. The following description results:

| <Concept_System> | | | |
|---|---|---|---|
| | **notation** | **referencedConceptSystem** | **importedConceptSystem** |
| RDF | | | |
| RDFS | | | RDF |
| XSD | | RDFS | |
| OWL | | RDFS | XSD |

| <Concept> (*excluding Relations*) | |
|---|---|
| | **container** |
| rdf:Property | RDF |
| rdfs:Resource | RDFS |
| rdfs:Class | RDFS |
| rdfs:Datatype | RDFS |
| rdfs:Literal | RDFS |
| xsd:string | XSD |
| xsd:decimal | XSD |
| xsd:integer | XSD |
| xsd:boolean | XSD |
| xsd:date | XSD |
| *...other XSD datatypes...* | |
| owl:Class | OWL |
| owl:Thing | OWL |
| owl:Nothing | OWL |
| owl:ObjectProperty | OWL |
| owl:DatatypeProperty | OWL |

| <Binary_Relation> | | | | | |
|---|---|---|---|---|---|
| | **container** | **role** | **reflexivity** | **symmetry** | **transitivity** |
| instance-type | RDF | instance<br>rdf:type | | asymmetric | intransitive |
| role-range | RDFS | role<br>rdfs:range | | asymmetric | intransitive |
| class-subsumption | RDFS | subclass<br>rdfs:subclassOf | | asymmetric | transitive |
| property-subsumption | RDFS | subproperty<br>rdfs:subpropertyOf | | asymmetric | transitive |
| class-equivalence | OWL | owl:equivalentClass | | symmetric | transitive |
| property-equivalence | OWL | owl:equivalentProperty | | symmetric | transitive |
| individual-equivalence | OWL | owl:sameAs | | symmetric | transitive |
| inequality | OWL | owl:differentFrom | | symmetric | intransitive |

| | | | | | |
|---|---|---|---|---|---|
| disjointness | OWL | owl:disjointFrom | | symmetric | intransitive |
| complementarity | OWL | owl:complementOf | | symmetric | intransitive |

| <Relation> (*excluding Binary_Relations*) | | | |
|---|---|---|---|
| | **container** | **arity** | **role** |
| owl:AllDifferent | OWL | | operand |
| owl:allValuesFrom | OWL | 3 | class |
| | | | role |
| | | | range |
| owl:someValuesFrom | OWL | 3 | class |
| | | | role |
| | | | range |
| owl:hasValue | OWL | 3 | class |
| | | | role |
| | | | value |
| owl:intersectionOf | OWL | | intersection |
| | | | operand |
| owl:unionOf | OWL | | union |
| | | | operand |
| owl:oneOf | OWL | | enumeration |
| | | | member |

| <Relation_Role> | | |
|---|---|---|
| | **multiplicity** | **ordinal** |
| instance | | 1 |
| rdf:type | | 2 |
| subclass | | 1 |
| rdfs:subclassOf | | 2 |
| subproperty | | 1 |
| rdfs:subpropertyOf | | 2 |
| role | | 1 |
| rdfs:range | | 2 |
| owl:equivalentClass | | |
| owl:equivalentProperty | | |

| | | |
|---|---|---|
| owl:sameAs | | |
| owl:differentFrom | | |
| owl:disjointFrom | | |
| owl:complementOf | | |
| owl:AllDifferent.operand | | |
| owl:allValuesFrom.class | | 1 |
| owl:allValuesFrom.role | | 2 |
| owl:allValuesFrom.range | | 3 |
| owl:someValuesFrom.class | | 1 |
| owl:someValuesFrom.role | | 2 |
| owl:someValuesFrom.range | | 3 |
| owl:hasValue.class | | 1 |
| owl:hasValue.role | | 2 |
| owl:hasValue.value | | 3 |
| owl:intersectionOf.intersection | | 1 |
| owl:intersectionOf.operand | | 2 |
| owl:unionOf.union | | 1 |
| owl:unionOf.operand | | 2 |
| owl:oneOf.enumeration | | 1 |
| owl:oneOf.member | | 2 |

| <Link> | | | |
|---|---|---|---|
| **container** | **relation** | **link_end** | |
| | | **end_role** | **end** |
| RDF | instance-type | instance | rdf:type |
| | | rdf:type | rdf:Property |
| RDFS | instance-type | instance | rdfs:Class |
| | | rdf:type | rdfs:Class |
| RDFS | instance-type | instance | rdfs:range |
| | | rdf:type | rdf:Property |
| RDFS | role-range | role | rdfs:range |
| | | rdf:range | rdf:Class |
| RDFS | role-range | role | role |
| | | rdf:range | rdf:Property |

| | | | |
|---|---|---|---|
| RDFS | role-range | role | rdf:type |
| | | rdf:range | rdfs:Class |
| RDFS | instance-type | instance | rdfs:subclassOf |
| | | rdf:type | rdf:Property |
| RDFS | role-range | role | rdfs:subclassOf |
| | | rdf:range | rdfs:Class |
| RDFS | role-range | role | subclass |
| | | rdf:range | rdfs:Class |
| RDFS | instance-type | instance | rdfs:subpropertyOf |
| | | rdf:type | rdf:Property |
| RDFS | role-range | role | rdfs:subpropertyOf |
| | | rdf:range | rdf:Property |
| RDFS | role-range | role | subproperty |
| | | rdf:range | rdf:Property |
| RDFS | instance-type | instance | rdfs:Resource |
| | | rdf:type | rdfs:Class |
| RDFS | class-subsumption | subclass | rdfs:Class |
| | | rdfs:subclassOf | rdfs:Resource |
| RDFS | instance-type | instance | rdf:Property |
| | | rdf:type | rdfs:Class |
| RDFS | class-subsumption | subclass | rdf:Property |
| | | rdfs:subclassOf | rdfs:Resource |
| RDFS | instance-type | instance | rdfs:Datatype |
| | | rdf:type | rdfs:Class |
| RDFS | class-subsumption | subclass | rdfs:Datatype |
| | | rdfs:subclassOf | rdfs:Class |
| RDFS | instance-type | instance | rdfs:Literal |
| | | rdf:type | rdfs:Class |
| RDFS | class-subsumption | subclass | rdfs:Literal |
| | | rdfs:subclassOf | rdfs:Resource |
| XSD | instance-type | instance | xsd:string |
| | | rdf:type | rdfs:Datatype |
| XSD | instance-type | instance | xsd:decimal |
| | | rdf:type | rdfs:Datatype |
| XSD | instance-type | instance | xsd:integer |

| | | | |
|---|---|---|---|
| | | rdf:type | rdfs:Datatype |
| XSD | instance-type | instance | xsd:boolean |
| | | rdf:type | rdfs:Datatype |
| XSD | instance-type | instance | xsd:date |
| | | rdf:type | rdfs:Datatype |
| *...other XSD datatype links...* | | | |
| OWL | instance-type | instance | owl:Thing |
| | | rdf:type | owl:Class |
| OWL | instance-type | instance | owl:Nothing |
| | | rdf:type | owl:Class |
| *...other OWL metamodel links...* | | | |

### F.4.2 Car Registration Ontology

An OWL ontology has been developed for illustration, for the application described in ISO TR 9007 Appendix B. Below is a graphical depiction of the ontology, as rendered with OntoViz.

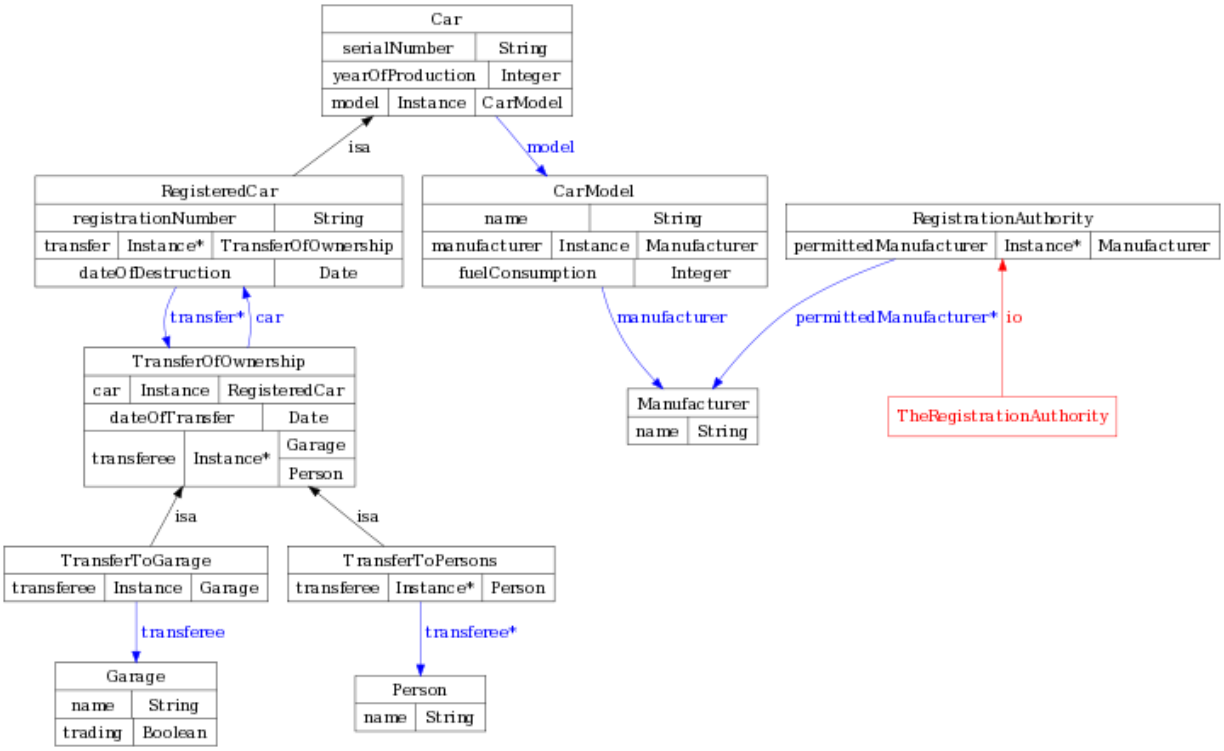



**Figure F-2 — Car Registration Ontology**

Below is the complete text of the ontology, in Turtle syntax:

```
@prefix :  <http://xmdr.org/ont/ISO9007.owl#> .
```

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .

<http://xmdr.org/ont/ISO9007.owl>
      rdf:type owl:Ontology .

:Car
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :yearOfProduction
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :model
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :serialNumber
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
               :Manufacturer , :RegistrationAuthority , :CarModel .

:CarModel
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :fuelConsumption
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :manufacturer
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
               :Manufacturer , :Car , :RegistrationAuthority .

:Manufacturer
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
               :Car , :RegistrationAuthority , :CarModel .
```

```
:RegistrationAuthority
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "5"^^xsd:int ;
                owl:onProperty :permittedManufacturer
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
               :Manufacturer , :Car , :CarModel ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf (:TheRegistrationAuthority)
              ] .

:TheRegistrationAuthority
      rdf:type :RegistrationAuthority .

:RegisteredCar
      rdf:type owl:Class ;
      rdfs:subClassOf :Car ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :registrationNumber
              ] .

:TransferOfOwnership
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:minCardinality "1"^^xsd:int ;
                owl:onProperty :transferee
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :dateOfTransfer
              ] ;
      owl:disjointWith :Person , :Garage , :Manufacturer , :Car ,
               :RegistrationAuthority , :CarModel ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:unionOf (:TransferToGarage :TransferToPersons)
              ] .

:TransferToGarage
      rdf:type owl:Class ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :transferee
              ] ;
      owl:disjointWith :TransferToPersons ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:allValuesFrom :Garage ;
                            owl:onProperty :transferee
                          ] :TransferOfOwnership)
```

```
              ] .

:Garage
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :trading
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Manufacturer ,
               :Car , :RegistrationAuthority , :CarModel .

:TransferToPersons
      rdf:type owl:Class ;
      owl:disjointWith :TransferToGarage ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:intersectionOf      (:TransferOfOwnership      [      rdf:type
owl:Restriction ;
                            owl:allValuesFrom :Person ;
                            owl:onProperty :transferee
                          ])
              ] .

:Person
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      owl:disjointWith :TransferOfOwnership , :Garage , :Manufacturer ,
               :Car , :RegistrationAuthority , :CarModel .

:yearOfProduction
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :Car ;
      rdfs:range xsd:int .

:model
      rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
      rdfs:domain :Car ;
      rdfs:range :CarModel .

:serialNumber
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :Car ;
      rdfs:range xsd:string .

:name
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain
              [ rdf:type owl:Class ;
                owl:unionOf (:CarModel :Manufacturer :Garage :Person)
```

```
              ] ;
      rdfs:range xsd:string .

:fuelConsumption
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:comment
            "number of litres of hydrocarbon fuel per 100 kilometres. Ranges from
4 to 25."^^xsd:string ;
      rdfs:domain :CarModel ;
      rdfs:range xsd:int .

:manufacturer
      rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
      rdfs:domain :CarModel ;
      rdfs:range :Manufacturer .

:permittedManufacturer
      rdf:type owl:ObjectProperty ;
      rdfs:domain :RegistrationAuthority ;
      rdfs:range :Manufacturer .

:registrationNumber
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :RegisteredCar ;
      rdfs:range xsd:string .

:dateOfDestruction
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :RegisteredCar ;
      rdfs:range xsd:date .

:transfer
      rdf:type owl:ObjectProperty , owl:InverseFunctionalProperty ;
      rdfs:domain :RegisteredCar ;
      rdfs:range :TransferOfOwnership ;
      owl:inverseOf :car .

:car
      rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
      rdfs:domain :TransferOfOwnership ;
      rdfs:range :RegisteredCar ;
      owl:inverseOf :transfer .

:transferee
      rdf:type owl:ObjectProperty ;
      rdfs:domain :TransferOfOwnership ;
      rdfs:range
              [ rdf:type owl:Class ;
                owl:unionOf (:Garage :Person)
              ] .

:dateOfTransfer
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :TransferOfOwnership ;
      rdfs:range xsd:date .

:trading
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :Garage ;
      rdfs:range xsd:boolean .
```

EDITOR'S NOTE #55. '(Action required) This example is incomplete.

*<Description in terms of 11179-3 Concept System metamodel to follow here>*

# Bibliography

[1] ISO/TR 9007:1987, *Information processing systems — Concepts and terminology for the conceptual schema and the information base*

TR 9007 provides information on conceptual modelling.

[2] ISO/IEC 10027:1990, *Information technology — Information Resource Dictionary System (IRDS) framework*

ISO/IEC 10027 describes the concept of levels of modelling.

[3] ISO/IEC TR 10032:2003, *Information technology — Reference model for data management*

ISO/IEC 10032 describes the concept of levels of modelling.

[4] ISO/IEC 11179-4, Information technology — Metadata registries (MDR) — Part 4: Formulation of data definitions

[5] ISO/IEC 11179-5, Information technology — Metadata registries (MDR) — Part 5: Naming and identification principles

[6] ISO/IEC DIS 19501-1:200n, *Information technology — Unified Modeling Language (UML) Version 2.1.2 — Part 1: Infrastructure*

[7] ISO/IEC TR 20943-1 (2003), *Information technology — Achieving metadata registry content consistency — Part 1: Data elements*

TR 20943-1 provides guidelines for recording data elements in an ISO/IEC 11179-3 metadata registry.

[8] ISO/IEC TR 20943-3 (2004), *Information technology — Achieving metadata registry content consistency — Part 3: Value domains*

TR 20943-3 provides guidelines for recording value domains in an ISO/IEC 11179-3 metadata registry.

[9] ISO/IEC 24707:2007 Common Logic

Includes a specification of XCL.